APLICACIÓN HÍBRIDA MÓVIL DE MENSAJERÍA INSTANTÁNEA CON ESTEGANOGRAFÍA A TRAVÉS DE IMÁGENES PARA INCREMENTAR EL NIVEL DE CONFIDENCIALIDAD DE LA INFORMACIÓN TRANSMITIDA

ERIKA ROCIO SALINAS CHAVES
EDUARDO GONZÁLEZ JAIMES

UNIVERSIDAD PILOTO DE COLOMBIA
PROGRAMA DE INGENIERÍA DE SISTEMAS
PROYECTO
BOGOTÁ
2017

APLICACIÓN HÍBRIDA MÓVIL DE MENSAJERÍA INSTANTÁNEA CON ESTEGANOGRAFÍA A TRAVÉS DE IMÁGENES PARA INCREMENTAR EL NIVEL DE CONFIDENCIALIDAD DE LA INFORMACIÓN TRANSMITIDA

ERIKA ROCIO SALINAS CHAVES
EDUARDO GONZÁLEZ JAIMES

TRABAJO DE GRADO PARA OPTAR AL TÍTULO DE ESPECIALISTA EN SEGURIDAD INFORMÁTICA

ASESOR
CÉSAR IVÁN RODRÍGUEZ SÁNCHEZ

UNIVERSIDAD PILOTO DE COLOMBIA
PROGRAMA DE INGENIERÍA DE SISTEMAS
PROYECTO
BOGOTÁ
2017

Nota de aceptación:

_____
_____
_____
_____
_____
_____

_____
Firma del presidente del jurado

_____
Firma del jurado

_____
Firma del jurado

Bogotá, 18 de abril de 2017

CONTENIDO

pág.

LISTA DE TABLAS

pág.

LISTA DE CUADROS

pág.

LISTA DE FIGURAS

pág.

GLOSARIO

APACHE RIPPLE: un simulador que se ejecuta como una aplicación web en el explorador Google Chrome. En Cordova, este simulador puede servir para simular la aplicación en una serie de dispositivos iOS y Android, y proporciona compatibilidad básica con los complementos principales de Cordova, como geolocalización y orientación del dispositivo.[1]

ANDROID: es un sistema operativo para dispositivos móviles. Está basado en GNU/Linux. Esta plataforma permite el desarrollo de aplicaciones por terceros (personas ajenas a Google), para lo cual, los desarrolladores deben escribir código gestionado en el lenguaje de programación Java y controlar los dispositivos por medio de bibliotecas desarrolladas o adaptadas por Google.[2]

COMPUTACIÓN EN LA NUBE: es un término general para denominar cualquier cosa que tenga que ver con la provisión de servicios de hospedaje a través de Internet. Estos servicios se dividen en tres grandes categorías: infraestructura como servicio (*IaaS*), plataforma como servicio (*PaaS*) y software como servicio (*SaaS*). El nombre de computación en la nube fue inspirado por el símbolo de nube, que se utiliza a menudo para representar a Internet en imágenes y diagramas de flujos.[3]

CONFIDENCIALIDAD: propiedad de la información de no estar disponible o ser divulgada a personas, procesos o entidades no autorizados.[4]

INFORMACIÓN: conjunto de datos procesados a cerca de un hecho, fenómeno o suceso que permite precisar, definir o explicar su conocimiento.[5]

MICROSOFT AZURE: Es la plataforma de computación en nube pública de Microsoft. Proporciona una gama de servicios en la nube, incluidos los de computación, analítica, almacenamiento y redes. Los usuarios pueden elegir entre

[1] MICROSOFT. MSDN Library. Simulador Apache Ripple. {En línea} {Febrero de 2017}. Disponible en https://msdn.microsoft.com/es-es/library/dn757052.aspx
[2] GSMSPAIN. {En línea}. {enero de 2017}. Disponible en http://www.gsmspain.com/glosario
[3] ROUSE, Margaret. Search DataCenter. {En línea}. {Febrero de 2017}. Disponible en http://searchdatacenter.techtarget.com/es/definicion/Microsoft-Azure-Windows-Azure
[4] INTERNATIONAL ORGANIZATION FOR STANDARIZATION. (2014). ISO/IEC 27000.
[5] Ibid.

estos servicios para desarrollar y escalar nuevas aplicaciones, o ejecutar aplicaciones existentes en la nube pública.[6]

NTC ISO/IEC 27001: norma que establece los requisitos para un sistema de gestión de la seguridad de la información (SGSI). Primera publicación en 2005; segunda edición en 2013. Es la norma en base a la cual se certifican los SGSI a nivel mundial.

SEGURIDAD INFORMÁTICA: procesos, actividades, mecanismos que consideran las características y condiciones de sistemas de procesamiento de datos y su almacenamiento con el fin de garantizar su confidencialidad, integridad y disponibilidad. [7]

SEGURIDAD DE LA INFORMACIÓN (*Information security*): preservación de la confidencialidad, integridad y disponibilidad de la información. [8]

[6] ROUSE, Op. cit.
[7] INTERNATIONAL ORGANIZATION FOR STANDARIZATION, Op. cit.
[8] Ibid.

INTRODUCCIÓN

El uso de los dispositivos móviles crece a tasas exponenciales, diversos estudios ubican este crecimiento en el último año entre un 40% y un 50%. Según cifras del boletín trimestral de las TIC, en Colombia se estimó que más de 58 millones de personas llegaron a tener un teléfono móvil al terminar el tercer trimestre de 2016 y el crecimiento de uso de estos dispositivos móviles es de 120,1% en el país, con relación al mismo trimestre del año anterior[9]. A medida que estas cifras aumentan, el uso de llamadas de celular, mensajes por correo y mensajes de texto disminuye. El tener a la mano un dispositivo inteligente, llámese teléfono o tableta, y una conexión a Internet, ya sea a través de datos o por red inalámbrica, permite que los usuarios se puedan comunicar de una manera rápida y directa con sus contactos o personas agregadas.

Aplicaciones como *Facebook Messenger*, *WhatsApp*, *Telegram* y *Line*, entre otras, abarcan grandes cuotas de mercado entre los usuarios, ofreciendo beneficios y funcionalidades a bajo costo. Un usuario que quiera comunicarse con sus contactos del teléfono o de correo, sólo debe descargar alguna de estas aplicaciones de la tienda a la que se encuentra registrado e inmediatamente estará en capacidad de enviar mensajes a otros usuarios que ya la hayan instalado. Un esquema relativamente sencillo, pero que deja algunas inquietudes frente a la preocupación de los usuarios con los datos que se van a enviar a través de ellas. ¿Se está seguro que el mensaje enviado a un usuario en particular sólo es recibido por él? Si se utilizan estas herramientas en entornos laborales, ¿la información sensible que se envía a otros colegas, jefes, etc. sólo va a ser observada por ellos? Y, en caso ser así, ¿cómo se asegura que luego de haber sido recibido el mensaje, este no va a ser observado por alguien diferente al receptor cuando se deja el teléfono fuera de vista o en caso de una pérdida o robo?

El presente proyecto, busca construir una aplicación de mensajería instantánea para dispositivos móviles con sistemas operativos *Android* y *iOS*, que incluya un módulo de seguridad basado en esteganografía, un método para ocultar información por medio de archivos como imágenes, y que permitirá que los usuarios intercambien información sin descuidar la confidencialidad de esta, agregando un nivel de seguridad mayor al que brindan las aplicaciones de mensajería actuales.

[9] MinTic. {En línea}. 3 de Enero de 2017. *Boletín trimestral del sector TIC. Cifras tercer semestre de 2016.* Obtenido de MinTIC. Gobierno de Colombia: http://colombiatic.mintic.gov.co/602/w3-article-47512.html

JUSTIFICACIÓN

En el contexto actual, las noticias relativas a información filtrada entre gobiernos, grandes estafas a corporaciones por mensajes entre sus ejecutivos, distribución de mensajes, fotografía y videos de personas famosas son pan de cada día y se convierten en un tema que ya no sorprende, incluso conociendo que las herramientas que son vulneradas, son las mismas que se utilizan día a día. Ante la necesidad de confidencialidad de la información que se transmite a través de las aplicaciones de mensajería existentes, es importante aplicar mecanismos que brinden seguridad sobre los datos que se intercambian entre los usuarios.

Construir una aplicación de mensajería, con un agregado de esteganografía y criptografía a estos mensajes, permitirá a los usuarios interesados en proteger sus datos, disponer de una herramienta de comunicación directa con sus contactos sin descuidar la protección de su información.

Se busca brindar confidencialidad a la información que se transmite a través de mensajes, ya que en las aplicaciones comúnmente utilizadas los datos son planos y visibles para cualquier persona que tenga acceso al dispositivo utilizado. Más aún, cuando el derecho fundamental a la intimidad y a la privacidad de los datos personales es una obligación y hace parte de la legislación de cada país.

1 PLANTEAMIENTO DEL PROBLEMA

1.1 DESCRIPCIÓN DEL PROBLEMA

En la actualidad se tienen diversas herramientas de mensajería instantánea usadas día a día, tanto a nivel personal, como laboral e incluso comercial, ya que varias empresas comienzan a girar sus estrategias de mercadeo y contacto a usuarios, a través de estas herramientas.

Frente al uso masivo y cotidiano de parte de los usuarios, es importante indagar acerca de la confidencialidad que pueden tener los datos enviados a través de todo el flujo de información, desde el momento en que sale del dispositivo del usuario emisor, hasta el momento en el que el usuario receptor lo recibe e incluso tiempo después, ya que esta información sigue disponible en los históricos de mensajes en el dispositivo o en los servidores de la aplicación utilizada.

1.2 FORMULACIÓN DEL PROBLEMA

¿Es posible implementar una aplicación móvil de mensajería instantánea que brinde al usuario una mayor seguridad en cuanto a la confidencialidad e integridad de sus mensajes frente a las utilizadas actualmente?

1.3 TIPO DE ESTUDIO INVESTIGATIVO

Descriptivo. Mediante el desarrollo de una aplicación de mensajería instantánea con un sistema criptográfico y esteganografía de imágenes, se busca brindar un nivel de confidencialidad entre los usuarios que intercambian información por medio de dispositivos móviles, ya que las aplicaciones de mensajería actual, no la brindan, de esta manera se plantea una respuesta y posible solución a partir de los conocimientos adquiridos.

2 OBJETIVOS

2.1 GENERAL

Construir una aplicación de mensajería instantánea para dispositivos móviles que ofrezca confidencialidad a la información transmitida entre las dos personas participantes.

2.2 ESPECÍFICOS

- Definir criterios que permitan establecer los parámetros para la elaboración de una aplicación de mensajería instantánea, utilizando los conocimientos adquiridos durante la especialización.

- Diseñar la arquitectura del sistema de mensajería con el método de criptografía y esteganografía definido.

- Implementar una aplicación híbrida para dispositivos móviles que brinde confidencialidad a la información que se transmite a través de un algoritmo esteganográfico.

3 ALCANCE Y LIMITACIONES

3.1 ALCANCE

Desarrollar una aplicación hibrida para dispositivos móviles en sistemas operativos Android y iOS, que permita establecer una conversación punto a punto por medio del intercambio de mensajes que se transmiten a través de un método esteganográfico que oculta la información en imágenes. Por tal motivo, se han implementado medidas de seguridad para el desarrollo del software buscando la preservación de los principios de confidencialidad e integridad de la información.

3.2 LIMITACIONES

Las limitaciones se encuentran alineadas con las actividades planeadas para el desarrollo de la aplicación, puesto que está implementada con fines académicos, estas son:

- No se publicará en las tiendas oficiales *Play Store* (Android) o *App Store* (iOS), ya que es una versión académica. Esta se podrá instalar y ejecutar, a partir de archivos que se entreguen por medio de repositorios.

- No se implementa una funcionalidad de conversaciones grupales, únicamente punto a punto, por lo que la conversación se podrá hacer sólo de usuario a usuario.

- Se tienen varios parámetros configurados a nivel de la aplicación. En una versión de producción y/o comercial, estos se podrían cambiar para que sea el usuario final quien los personalice de acuerdo a sus requerimientos o consideraciones de uso.

4 MARCO TEÓRICO

4.1 PLATAFORMA DE DESARROLLO DE APLICACIONES HÍBRIDAS

4.1.1 Comparación de alternativas. En la actualidad se encuentran tres enfoques generales de desarrollo en el momento de construir una aplicación móvil. Cada uno tiene sus ventajas y desventajas frente a los demás y con el paso del tiempo han disputado liderazgo de la alternativa más utilizada, teniendo como casos de éxito para mostrar, un conjunto de aplicaciones que han alcanzado un gran número de descargas y de desarrolladores que las usan y respaldan.

Las alternativas son:

- Aplicaciones nativas: desarrolladas de forma individual para cada una de las plataformas o sistemas operativos existentes en dispositivos móviles, manejando un lenguaje propio para cada una: Java para Android, Objetive-C para iOS, .NET.

- Aplicaciones híbridas: desarrolladas en diferentes lenguajes, dependiendo de la herramienta utilizada (normalmente HTML, JavaScript y CSS). Son implementadas sobre un *framework* o plataforma de desarrollo que posteriormente permite la generación de instaladores propios como aplicación nativa para cada una de las diferentes plataformas.

- Aplicaciones web: desarrolladas para su ejecución sobre un navegador, no son aplicaciones como tal, ya que son desplegadas sobre un servidor web y no necesitan ser instaladas en el dispositivo.

De acuerdo a una investigación realizada por IBM para sus sesiones de capacitación web[10], se puede comparar cada una de estas aplicaciones desde el punto de vista del tipo de desarrollo, teniendo como aspectos principales de estudio el acceso al dispositivo, su desempeño o velocidad de ejecución, el costo de desarrollo, la facilidad de distribución a través de tiendas de aplicaciones y el proceso de aprobación por parte del fabricante para que el producto final pueda ser incluido en una de estas tiendas. En la Tabla 1 se puede observar el resumen de estas comparaciones.

[10] IBM. White Papers. {En línea}. {Junio de 2016}. Native web or hybrid mobile app development. Worklight Webinar Series. Disponible en: ftp://public.dhe.ibm.com/software/pdf/mobile-enterprise/WSW14182USEN.pdf

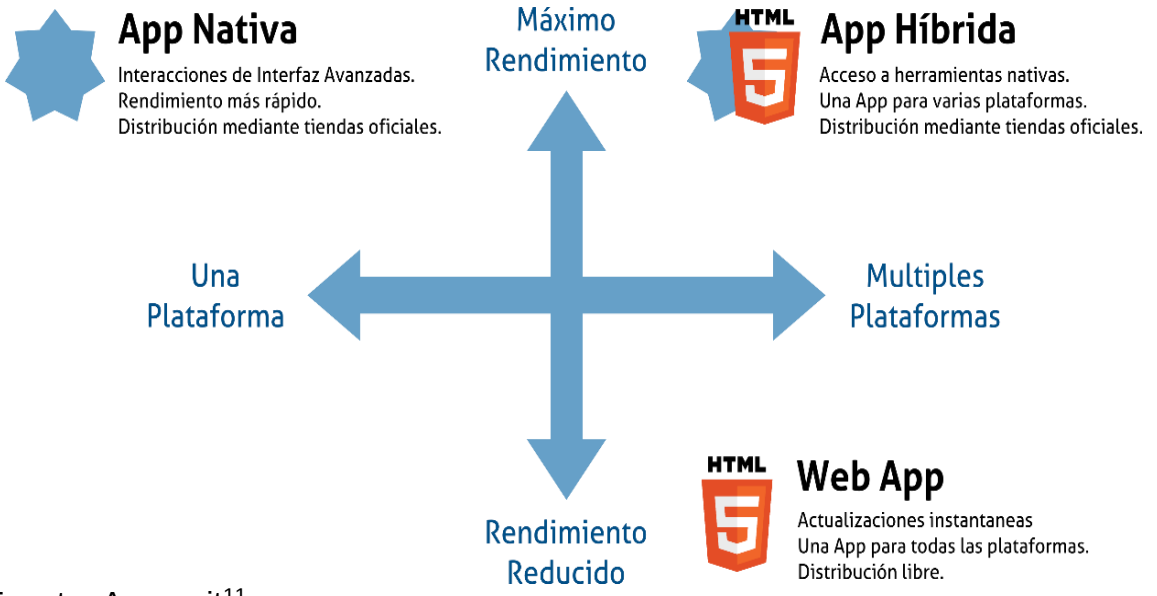
Tabla 1. Comparación de entornos de desarrollo de aplicaciones

Tipo de aplicación	Acceso al dispositivo	Velocidad	Costo de desarrollo	Tienda de aplicaciones	Proceso de aprobación
Nativa	Completo	Muy rápida	Costosa	Disponible	Obligatorio
Híbrida	Completo	Cercana a la nativa	Razonable	Disponible	Indirecto
Web	Parcial	Rápida	Razonable	No disponible	Ninguno

Fuente: Autores del proyecto

En la Figura 1, se comparan estos tres entornos de desarrollo en una matriz, la cual permite entender las fortalezas y debilidades de cada uno frente a su interoperabilidad y rendimiento:

Figura 1. Comparativa de enfoques para dispositivos móviles



Fuente: Accensit[11]

[11] ACCENSIT. Blog Accensit. {En línea}. {Julio de 2016} Obtenido de http://www.accensit.com/index.php/en/accensit-blog-en/150-mobileplatforms.html

4.1.2 Aplicaciones híbridas y herramientas de desarrollo. Una aplicación híbrida se encuentra justo en el medio entre una aplicación nativa y una web. Tienen la ventaja de que, en determinado momento, pueden hacer uso de funcionalidades nativas por intermedio de APIs para el acceso a características que dependen de cada sistema operativo como el acelerómetro, la cámara, el estado de batería, la geolocalización, etc.

Entre las herramientas más conocidas que permiten el desarrollo de este tipo de aplicaciones se encuentra Apache Cordova (con su distribución PhoneGap), Titanium Appcelerator y Xamarin como las líderes del mercado, tal como se señala en la Tabla 2, un estudio realizado por Infotech[12]:

Tabla 2. Herramientas de desarrollo de aplicaciones híbridas

Característica	**Apache Cordova (Phonegap)**	**Titanium**	**Xamarin**
Distribución	Android iOS Windows Phone BlackBerry	Android iOS BlackBerry	Android iOS Windows Phone
Lenguaje	HTML5 CSS JavaScript	JavaScript	C#
Código abierto	Si	Si	Si*
Interfaz	Web	Nativa	Nativa
Acceso a APIs del dispositivo	Si	No	No
Soporte DOM	No	Si	Si
Desempeño nativo	No	Si	Si

Fuente: Cygnet Infotech

[12] INFOTECH. Blog Cygnet Infotech. {En línea}. {Junio de 2016}. Obtenido de http://www.cygnetinfotech.com/blog/phonegap-or-titanium-or-xamarin-which-cross-platform-should-you-choose

(*) Xamarin fue adquirida en 2016 por Microsoft y se liberó su código, por lo que se debe considerar ahora como una herramienta de código abierto

4.2 MENSAJERÍA INSTANTÁNEA

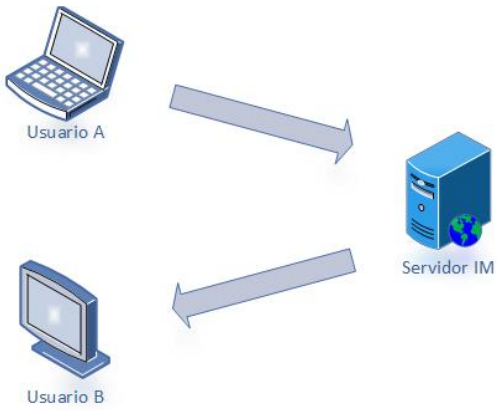
La mensajería instantánea es la forma de comunicación por medio de texto entre dos o más personas utilizando aplicaciones instaladas en sus computadores, dispositivos móviles, o incluso accediendo desde páginas web. A estas herramientas para poder acceder y enviar o recibir mensajes se les conoce como clientes. Estos clientes se conectan a uno o varios servidores, normalmente centralizados, que brindan el servicio, luego de una autenticación y revisión de contactos o personas con las que se puede comunicar, diferenciándose del correo electrónico tradicional en que las conversaciones se hacen en tiempo real.

Los sistemas de mensajería más utilizados actualmente son:

- WhatsApp
- Facebook Messenger
- Skype
- Line
- Snapchat
- Telegram
- Viber
- WeChat

En la Figura 2, se puede ver la arquitectura de un sistema de mensajería básica, que consiste en al menos dos clientes conectados a un servidor por medio de Internet. El usuario A envía un mensaje que es recibido por el servidor, el cual valida los datos del mensaje y lo reenvía al usuario B, también conectado a través de un cliente y quién será el receptor del mensaje.

Figura 2. Arquitectura básica de un sistema de mensajería instantánea



Fuente: Autores del proyecto.

4.2.1 Protocolos de mensajería. Entre los protocolos de mensajería más utilizados se encuentran:

• Skype: protocolo de código cerrado, que inicialmente se desarrolló en el lenguaje de programación Pascal. Para el cifrado de la transferencia de datos o mensajería instantánea y llamadas telefónicas, Skype utiliza AES (*Advanced Encryption Standard*).

• IRC (*Internet Relay Chat*): protocolo que se basa en la arquitectura cliente-servidor y es adecuado para funcionar en varias máquinas de un modo distribuido. Para comenzar una charla, los usuarios deben utilizar una aplicación cliente con un alias antes de conectarse, luego este software se conecta a un servidor en el que funciona una aplicación INCD (*IRC Daemon* o servidor IRC) el cual gestiona los canales de comunicación y las conversaciones.

• OSCAR (*Open System for Communication in Realtime*): es de desarrollo propietario por tal motivo no se puede acceder a la documentación ni al código de desarrollo. La comunicación entre dos usuarios no se realiza de forma directa, sino a través de múltiples servidores centrales BOS (*Basic OSCAR SERVICE*) y un servidor de autorizaciones, que se encargan de la entrega y recepción de los mensajes a sus destinatarios.

• XMPP (*Extensible Messaging and Presence Protocol*): protocolo abierto y extensible, para el intercambio de datos utiliza una plataforma XML. Su funcionamiento topológico se basa en la arquitectura cliente-servidor y, mediante TLS permite cifrar los mensajes empleando diferentes algoritmos como RSA y DSS.

• WebSocket (producto SignalR): Librería que permite la comunicación en tiempo real para entornos .NET utilizando conexiones persistentes y concentradores (*hubs*).

4.3 SIGNALR

Hoy en día las comunicaciones inalámbricas por medio de dispositivos móviles y el acceso a internet de banda ancha, ha tenido un aumento exponencial de uso entre los usuarios, esto genera que las mismas se realicen en tiempo real. Las aplicaciones más populares que se caracterizan por este tipo de comunicación son las de redes sociales, servicios de ayuda en línea, banca y entidades financieras y sitios de comercio electrónico, entre otras.

Realizar este tipo de comunicación en la web normalmente no es sencillo, ya que la mayoría de protocolos que la sustentan, están basados en un modelo cliente y un servidor síncrono: uno o varios clientes realizan una conexión hacia el servidor y le transmiten una acción a realizar, éste la procesa y les retorna la respuesta, cerrándose la conexión de forma inmediata[13].

Este modelo de comunicación no es el ideal para este tipo de necesidades, ya que siempre el cliente debe estar enviando peticiones de conexión al servidor para indagar por nuevos mensajes. Por tal motivo se tienen herramientas como SignalR, que permiten realizar una comunicación en tiempo real, sencilla y bidireccional entre el servidor y el cliente. Esto quiere decir que no solo el cliente puede iniciar la comunicación con el servidor, sino que el servidor también puede ponerse en contacto con el cliente de forma asíncrona[14]. Este tipo de comunicación se puede observar en la Figura 3.

[13] AGUILAR, José. *Introducción a SignalR.* {En línea}. {Junio de 2016}. Disponible en http://www.variablenotfound.com/2012/03/signalr-iv-hubs.html

[14] APPEL, R. Usa SignalR para crear aplicaciones modernas. {En línea}. {Junio de 2016}. Disponible en https://msdn.microsoft.com/es-es/magazine

Figura 3. Comunicación asíncrona utilizando SignalR

Fuente: Autores del proyecto.

La transferencia de mensajes no consiste sólo en respuestas desde el servidor por peticiones del cliente (*pull*), sino en auténticas llamadas de métodos del servidor al cliente (*push*). Es por esto que utilizar un tipo de comunicación como el mencionado a la hora de escribir aplicaciones y sitios web, es una opción válida y justificada, que permite comunicaciones activas en el software en tiempo real.

En SignalR se pueden utilizar dos capas de abstracción para la comunicación entre el cliente y el servidor:

- Conexiones persistentes: las de más bajo nivel que permiten manejar eventos o mecanismos simples para conexiones y desconexiones de clientes y comunicarse de forma bidireccional con ellos.

- *Hubs*: modelo de alto nivel que ofrece una abstracción aún mayor, con una comunicación entre el cliente y el servidor de forma casi inmediata.

4.4 SQL SERVER

La información es uno de los activos de información más importantes para las organizaciones y/o empresas hoy en día y en consecuencia, requiere un adecuado tratamiento para salvaguardarla y minimizar los riesgos que puedan llegar a afectar su integridad, confidencialidad y disponibilidad. Por tal motivo, es indispensable almacenar la información digital en un solo repositorio, donde este integrada, organizada y relacionada entre sí, para que luego pueda ser utilizada y encontrada fácilmente. A esto se le conoce como "base de datos".

Las principales características de las bases de datos son: independencia lógica y física de los datos, redundancia mínima, integridad de los datos, consultas complejas para encontrar información, seguridad de acceso, respaldo y recuperación, acceso a través de lenguajes de programación estándar, entre otros[15]

De acuerdo a lo anterior, las bases de datos requieren de un sistema de información que las manejen y puedan ser fáciles de administrar. Existen varios sistemas, como Oracle, MySQL, PostgreSQL, MariaDB y SQL Server. Este último es uno de los más utilizados, ya que es un sistema que está enfocado en tener un mayor rendimiento, obtención, transacciones, consultas y análisis de la información que se encuentran de forma local o en la nube.

4.5 ANGULARJS

AngularJS es un *framework* de tipo modelo – vista - controlador (MVC) para JavaScript y de código abierto, que contiene un conjunto de librerías para el desarrollo y mantenimiento de aplicaciones web de una sola página[16]. Su principal ventaja es que adiciona funcionalidades y mejoras sobre el código HTML que escriben los programadores, permitiendo declarar vistas dinámicas en las aplicaciones y que estas sean fáciles de entender incluso para personas con conocimientos básicos en tecnología o programación. Algunos elementos que se encuentran dentro de AngularJS son:

- Modelo de la vista: es la capa donde se encuentra toda la información y/o datos, es decir, la que se encarga de hacer peticiones a las bases de datos para enviar o recibir información.

- Vistas: es el HTML y todo lo que representa datos o información, se trata del código que permite presentar los datos que el modelo proporciona.

- Controlador: es la capa que sirve de enlace entre la vista y el modelo, envía comandos al modelo para actualizar su estado y a la vista correspondiente para cambiar su presentación.

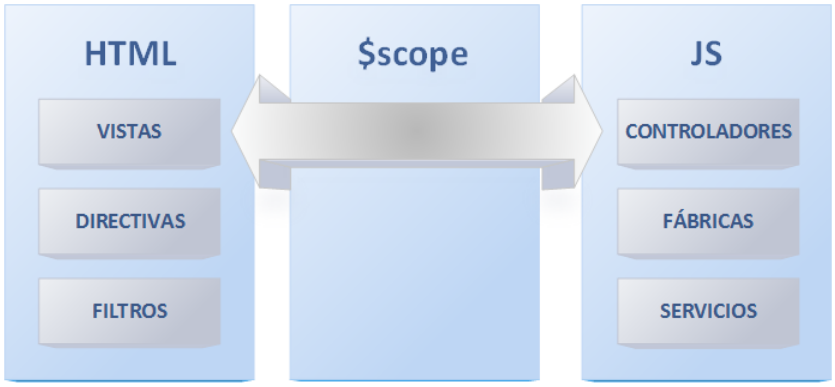
El *framework* se puede dividir en dos áreas, una es el HTML, es la parte declarativa, vistas, directivas y filtros que ofrece AngularJS, así como lo que realizan los

[15] Pérez, D. (Octubre de 2013). *Que son las bases de datos. .* Obtenido de http://www.maestrosdelweb.com/que-son-las-bases-de-datos

[16] Basalo, A. (28 de Agosto de 2014). *¿Qué es AngularJS?* Obtenido de http://www.desarrolloweb.com/articulos/que-es-angularjs-descripcion-framework-javascript-conceptos.html

programadores y la otra es el JavaScript como tal, en la cual se encuentran los controladores, factorías y servicios[17]. Esto se puede observar en la Figura 4.

Figura 4. Componentes del framework de AngularJS



Fuente: Autores del proyecto.

4.6 IONIC

Es una herramienta (capa) gratuita que trabaja sobre Cordova y permite desarrollar aplicaciones muy llamativas basadas en HTML gracias a AngularJS. El *framework* está construido con Angular y SASS (preprocesador CSS).

Sus principales características son:

- Alto rendimiento: por la mínima manipulación del DOM (modelo de objetos del documento), sin el uso de librerías como JQuery y con aceleraciones de transiciones por hardware. Ionic es una herramienta de velocidad rápida.

- Tiene una arquitectura central, robusta y seria para el desarrollo de aplicaciones, ya que utiliza AngularJS.

- Se inspira en las SDK de desarrollo móviles nativos más populares, por lo que permite desarrollar una vez y compilarla varias veces.

[17] Google. (Abril de 2016). *AngularJS by Google.* Obtenido de HTML enhanced for web apps: https://angularjs.org

• Ha sido diseñada para poder ser utilizada por cualquiera de los dispositivos móviles de hoy en día (iOS, Android, entre otros).

4.7 JQUERY

Considerado un *framework* de JavaScript, es un ambiente de desarrollo *open-source* que funciona en múltiples navegadores (Internet Explorer, Chrome, Mozilla Firefox, entre otros) y es compatible con CSS3. Su objetivo principal es hacer la programación del lado del cliente más fácil y sencilla, ya que se puede realizar muchas funciones de *script* frecuentes con menos líneas de código[18].

Las características de JQuery son:

• Agregar *plugins* (sub-sistema o componentes) fácilmente, ahorrando tiempo y esfuerzo.

• Flexible y rápido para el desarrollo web.

• Excelente integración con AJAX (JavaScript asíncrono y XML).

• Tiene una excelente comunidad de soporte

JQuery tiene una librería que puede ser utilizada con los dispositivos móviles: JQuery *Mobile*, que es un *framework web* optimizado para estos dispositivos, compatible con iOS, BlackBerry, Windows Mobile, Symbian y Android. Soporta temas de estilos y es ligero y rápido de utilizar.

4.8 APACHE CORDOVA

Apache Cordova es un *framework* para el desarrollo de aplicaciones móviles, que contiene un conjunto de APIs con acceso a algunas funciones del dispositivo, como la cámara, las imágenes, el acelerómetro, entre otros. De esta manera los programadores pueden desarrollar aplicaciones utilizando un mismo lenguaje,

[18] Mikoluk, K. (10 de Diciembre de 2013). *JQuery vs JavaScript: ¿Cuál es la Diferencia En Definitiva?* Obtenido de https://blog.udemy.com/jquery-vs-javascript-2-cual-es-la-diferencia-en-definitiva/

herramientas *web* genéricas como JavaScript, HTML5 y CSS3. Para tener como resulta aplicaciones hibridas que funcionen sobre diferentes sistemas[19].

Con Apache Cordova se pueden desarrollar aplicaciones móviles para las siguientes plataformas:

- Amazon Fire OS
- Android
- BlackBerry 10
- Firefox OS
- iOS
- Ubuntu
- Windows Phone
- Windows 8

Una misma aplicación desarrollada con Apache Cordova para todos los sistemas operativos móviles, puede tener cambios concretos y diferentes. Esto se realiza mediante la carpeta *merge* que tiene el *framework*, con la cual se tiene la posibilidad de incluir cualquier archivo (imágenes, estilos, *scripts*, entre otros) de manera que al compilar la aplicación se obtiene un instalador para cada plataforma, o un conjunto de iconos distinto, sin tener que escribir nada de código.

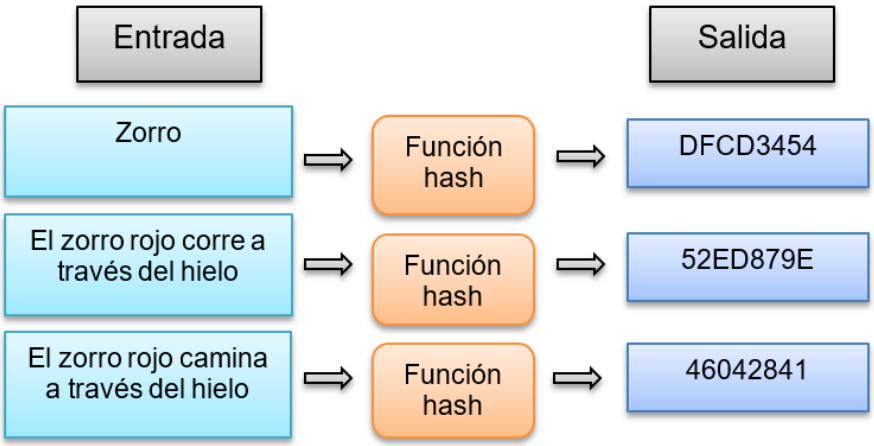
4.9 HASH

Es un algoritmo matemático que permite transformar una serie de datos de entrada que pueden ser texto, imágenes, archivos, contraseñas, entre otros, en una salida de datos de longitud fija. Esto se logra, mediante una función matemática que se aplica sobre ese conjunto de datos de entrada, y su salida es una huella digital, de tamaño fijo e independiente de la longitud de los datos de entrada.

[19] *Cordova.* (29 de Mayo de 2016). Obtenido de Documentation Cordova: https://cordova.apache.org/docs/es/latest/guide/overview

Sea *h* la función *hash*, y *x* algún flujo de bits de longitud cualquiera, la huella digital está dada por: *y = h(x)*. Esta huella digital es también conocida como *Message Digest* y es única para cada flujo de bits. En la Figura 5 se presentan varios ejemplos de esta función.

Figura 5. Ejemplos de cadenas de texto en función Hash



Fuente: Autores del proyecto.

Una de las características principales de este sistema criptográfico, es salvaguardar la integridad de la información transmitida en un canal probablemente inseguro, ya que permite verificar el valor *hash* (huella digital) de los datos recibidos con el valor *hash* (huella digital) de los datos que se enviaron, de esta manera se determina si la información es igual o existió alguna alteración en la misma.

Las propiedades de esta función son:

- Sea cual sea el mensaje de entrada, todos los *hashes* generados con una función *hash* tienen el mismo tamaño.

- No se puede reconstruir el mensaje original a partir de un *hash*.

- Es imposible generar un mensaje con un *hash* determinado, ya que un algoritmo de este tipo no es un algoritmo de encriptación.

- Mediante un *software* en línea es muy fácil generar un *hash*.

- Es poco probable encontrar un conjunto de datos diferentes con un mismo *hash* (aunque se han presentado pocos casos en el mundo con esta similitud).

Los tipos de algoritmos criptográficos de hash más utilizados son:

- MD5 (*Message Digest Algoritm 5* o algoritmo de firma de mensajes 5), es un algoritmo desarrollado por RSA *Data Security*, Inc., el cual toma un determinado tamaño a la entrada y genera un resultado de longitud fija de 128 bits. Este tipo de algoritmo no es seguro de utilizar actualmente, debido al aumento en la capacidad de procesamiento. En pruebas se han puesto de manifiesto algunas vulnerabilidades del algoritmo.

- SHA-1 (*Secure Hash Algorithm 1* o algoritmo de *hash* seguro 1), es muy parecido a MD5, pero toma como entrada un mensaje de longitud máximo 2^{64} bit y genera una huella digital de 160 bits en lugar de los 128 bits del MD5. Este tipo de algoritmo es más lento que el anterior, ya que el número de pasos para generar el *hash* son 80 frente a los 64 que toma MD5.

- SHA-2 (*Secure Hash Algorithm 2* o algoritmo de *hash* seguro 2), lleva algún tiempo posicionándose como el sucesor de SHA1, ya es utilizado en un número considerable de herramientas de seguridad y protocolos como lo son TLS, SSL, PGP, SSH, S/MINME, IPsec e incluso la moneda virtual *Bitcoin*. Este tipo de algoritmo toma como entrada un mensaje de longitud máximo de 2^{64} bit y genera una huella digital de 256 bits. El número de pasos para generar el hash es 64.

En el Cuadro 1, se enuncian las características del algoritmo *hash* SHA.

Cuadro 1. Tipos de algoritmo hash SHA

Tipo	**SHA-1**	**SHA-256**	**SHA-384**	**SHA-512**
Huella Digital	160 bits	256 bits	384 bits	512 bits
Mensaje	$<2^{64}$ bits	$<2^{64}$ bits	$<2^{128}$ bits	$<2^{128}$ bits
Bloque	512 bits	512 bits	1024 bits	1024 bits
Palabra	32 bits	32 bits	32 bits	32 bits
Etapas	80	64	80	80

Fuente: Autores del proyecto

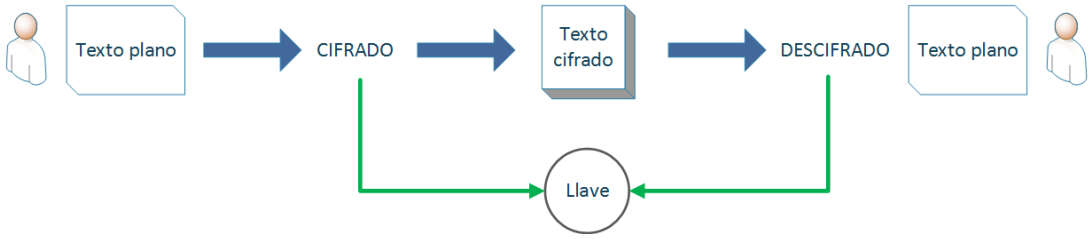
4.10 CRIPTOGRAFÍA

Esta palabra proviene de las griegas *kryptos* y *graphia*, que significan oculto y escritura, respectivamente. Es el procedimiento mediante el cual se busca un intercambio de información entre destinatario y receptor, alterándola mediante el uso de claves, para que no sea comprensible para intrusos que la observen.

4.10.1 Tipos de cifrado. Dependiendo de la forma de uso de estas claves, la criptografía se divide en dos tipos:

4.10.1.1 Criptografía simétrica. Se utiliza la misma clave para cifrar y descifrar el mensaje, por lo que debe ser conocida por los dos actores, lo cual plantea el problema de la transmisión de la misma, para iniciar una comunicación de forma segura. En la Figura 6 se observa un ejemplo de este tipo de criptografía.

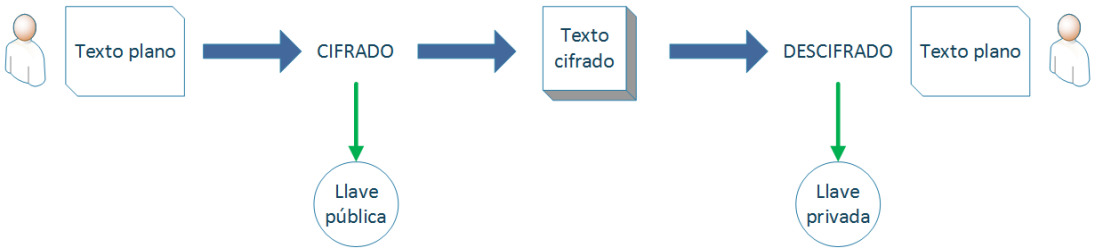
Figura 6. Criptografía simétrica



Fuente: Autores del proyecto

4.10.1.2 Criptografía asimétrica. Emplean dos claves conocidas como privada y pública, de tal forma que una se aplica en una función de cifrado y la otra en una posterior de descifrado del mensaje. La clave privada es conocida únicamente por su propietario y la pública por aquel que quiera hacerle llegar un mensaje. En la Figura 7 se puede observar un ejemplo de este tipo.

Figura 7. Criptografía asimétrica



Fuente: Autores del proyecto

4.10.2 Algoritmos de cifrado. Dentro de los algoritmos más usados en el cifrado de información se encuentran:

4.10.2.1 DES (Data Encryption Standard). Algoritmo de cifrado por bloques de 64 bits, aunque sólo se usan 56 de la misma (clave efectiva), consta de 16 iteraciones con operaciones *or exclusivo*, permutaciones y sustituciones. Actualmente se considera inseguro.

4.10.2.2 3DES (Triple DES). Extiende el tamaño de clave del algoritmo DES, aplicándolo tres veces con sucesión de claves diferentes, llegando a una longitud de 168 bits Es usado actualmente en diferentes dispositivos, pero está siendo reemplazado poco a poco por el algoritmo AES.

4.10.2.3 AES (Advanced Encryption Standard). También conocido como *Rijndael*, por la combinación de letras de los apellidos de sus autores (*Daemen* y *Rijmen*). Utiliza bloques de 128 bits y tamaños de llave de 128, 192 o 256. Es robusto y se puede atacar por fuerza bruta, aunque toma demasiada capacidad de procesamiento y tiempo, sobre todo para tamaños de clave grandes (256).

4.11 ESTEGANOGRAFÍA

La mensajería instantánea es la forma de comunicación por medio de texto entre dos o más personas utilizando aplicaciones instaladas en sus computadores, dispositivos móviles, o incluso accediendo desde páginas *web*. A estas herramientas para poder acceder y enviar o recibir mensajes se les conoce como clientes. Estos clientes se conectan a uno o varios servidores, normalmente centralizados, que brindan el servicio, luego de una autenticación y revisión de contactos o personas con las que se puede comunicar, diferenciándose del correo electrónico tradicional en que las conversaciones se hacen en tiempo real.

La esteganografía busca ocultar uno o varios mensajes dentro de otros, para que estos no se puedan observar fácilmente en el momento en que sean observados por alguien diferente al receptor. La palabra viene del griego *steganos* que significa oculto y *graphos* que significa escritura. Esta ciencia remonta desde el siglo XV y ha venido evolucionando a través de la historia desde los mensajes tallados en madera ocultos con cera hasta la utilización de medios digitales (imágenes, videos, audios, protocolos de comunicación, entre otros) en la actualidad con la ayuda de los avances tecnológicos de los últimos tiempos.

Por lo anterior, mediante la esteganografía se implementa un conjunto de técnicas para incrustar información en un objeto contenedor o portador, de esta forma, la misma pasa inadvertida para un tercero. Aunque este conozca el archivo, no sabe cuál fue el algoritmo utilizado para ocultar la información, ya que sólo lo debe saber el receptor del mensaje.

Maite Moreno, de Security ArtWork, enumera los actores implicados en un procedimiento esteganográfico[20]:

- El objeto contenedor o encubridor, que en la mayoría de los casos suele ser una imagen y que es utilizado para ocultar la información.

- El estego-objeto, que se trata del objeto contenedor junto con el mensaje encubierto.

- El adversario, serían todos aquellos actores externos, ya sean pasivos o activos, a los que se les quiere ocultar la información. Los adversarios pasivos sospechan que se puede estar llevando a cabo alguna transferencia de información encubierta en la comunicación e intentan descubrir el algoritmo para conocer el mensaje oculto, mientras que los activos además de intentar descifrar el algoritmo de comunicación, cambian la información contenida en el estego-objeto, de esta manera se afecta la integridad de la misma.

- El estego-análisis, es la ciencia que estudia la detección de información oculta.

4.11.1 Diferencias entre esteganografía y criptografía. Cada vez existen más intrusos activos y con niveles avanzados en tecnología donde su único objetivo es apoderase de la información. Debido a esto, la seguridad de la información ha incrementado sus métodos para salvaguardarla, por lo que el uso de la criptografía cada vez es más común hoy en día en las empresas de desarrollo de software

Tanto la esteganografía como la criptografía son campos distintos en su razón de ser, aunque ambas intentan proteger la información, la primera busca ocultar el mensaje en sí, de modo que no sea conocida su existencia o envío y la segunda lo asegura mediante su cifrado, para que no sea comprensible ante los ojos de un intruso que ignora las claves de cifrado o descifrado, incluso aunque sepa de la existencia de esa información.

[20] MORENO, M. (15 de Abril de 2010). Security ArtWork: Introducción a la esteganografía (I). {En línea}. {Julio de 2016}. Disponible en http://www.securityartwork.es

Mediante la combinación de la criptografía y la esteganografía, se pueden ocultar datos con un grado de seguridad sorprendentemente alto, ya que el mensaje a intercambiar se cifra (de forma robusta) y luego se introduce en un objeto contenedor. De esta forma, aunque un intruso descubra el patrón esteganográfico, tendrá dificultades adicionales en llegar a conocer el mensaje intercambiado[21].

La combinación de estas dos técnicas tiene otra ventaja adicional, ya que cuando se utiliza la criptografía en un intercambio y/o transferencia de mensajes, se está dando un punto de partida para que un atacante intente descubrir la información. Al implementar adicionalmente un método esteganográfico, se minimiza el riesgo de que se conozca la existencia de un mensaje privado, por lo que no sería perceptible para un intruso y lo más probable es que no pueda deducir que en el objeto contenedor existe un mensaje cifrado.

4.11.2 Técnicas conocidas en la esteganografía de imágenes. Entre las técnicas más conocidas y utilizadas para aplicar esteganografía a imágenes se tienen:

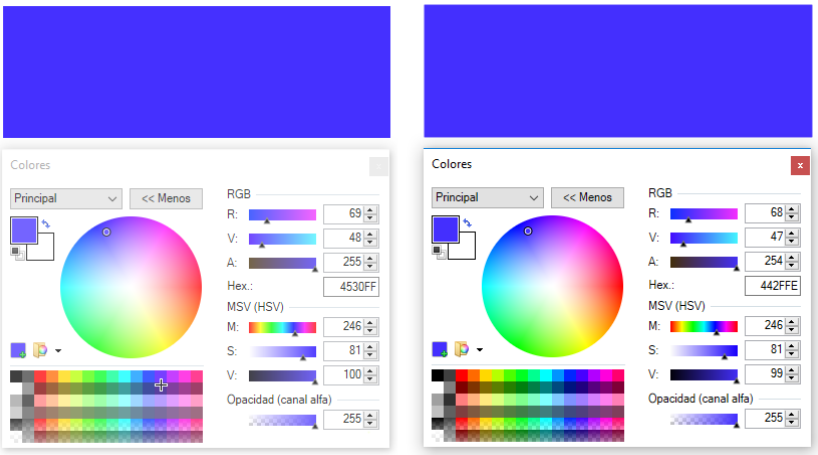
4.11.2.1 Sustitución de bits. Se utilizan bits propios del objeto contenedor para incluir la información a ocultar. Teniendo en cuenta que se saca un bit para reemplazarlo por otro, esta técnica no altera el tamaño del contenedor o canal del mensaje y, si se hace con eficacia y respetando las particularidades de su formato, no ocasionará gran impacto en su calidad.

Teniendo en cuenta el trabajo con imágenes, foco de este proyecto, se puede explicar esta técnica con el reemplazo de los bits en una escala de color determinada, modificando el código RGB de color para cada pixel a un tono mayor o menor, imperceptible para el ojo humano.

En la Figura 8, se pueden ver dos muestras de color, en el que se modificó el bit menos significativo sobre cada uno de los tres colores primarios, pasando el rojo (R) de 69 a 68, el verde (G) de 48 a 47 y el azul (B) de 255 a 254. Esto significa un cambio en el último bit de cada valor de 0 a 1 o de 1 a 0, según sea el caso. El resultado es prácticamente el mismo color, al menos para el ojo humano.

[21] MOURA, William. Esteganografía, el arte de ocultar información. {En línea}. {Julio de 2016} Disponible en http://www.egov.ufsc.br:8080/portal/conteudo/esteganograf%C3%ADa-el-arte-de-ocultar-informaci%C3%B3n

Figura 8. Colores con variaciones mínimas en su escala RGB



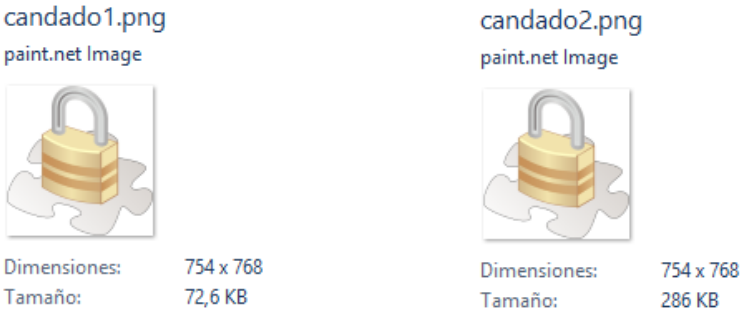
Fuente: Autores del proyecto.

4.11.2.2 Inserción de bits. Esta técnica consiste en añadir bits del mensaje que se quiere ocultar, en determinados sitios que no afectan la integridad del archivo contenedor como espacios, saltos de línea o espacios vacíos que se pueden crear.

Para la esteganografía de imágenes, una inserción puede consistir en agregar la cantidad de bits que tiene el mensaje a ocultar en un espacio entre la cabecera, que contiene los metadatos o características de la imagen y los datos de la imagen como tal, aprovechando una propiedad en la que se define el número de bits existentes entre dicha cabecera y la información de los pixeles de la imagen. Esta técnica tiene la desventaja de que el archivo se incrementa en tamaño tanto como ocupe el mensaje a ocultar, sin embargo, al no modificar la información original del archivo contenedor, se asegura que la calidad sigue siendo la misma a la de la imagen antes de insertar el mensaje.

Si se inserta un mensaje en un archivo de imagen de extensión PNG, como el mostrado en la Figura 9, se puede observar un crecimiento en su tamaño, aunque conserve las mismas dimensiones y calidad.

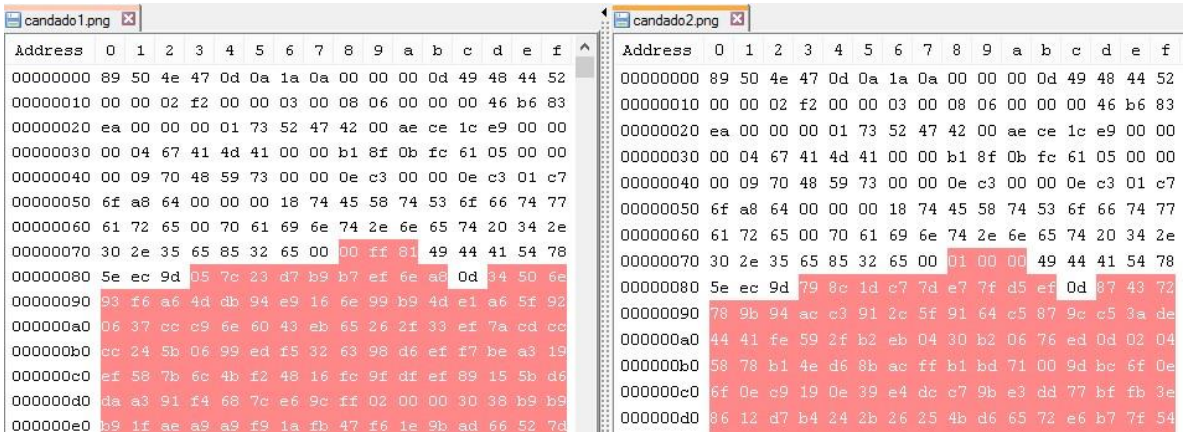
Figura 9. Comparación de imágenes original y contenedora del mensaje



Fuente: Autores del proyecto.

En la Figura 10 se visualiza como, al comparar estos dos archivos con un editor de texto hexadecimal, se observan las diferencias en las cadenas de bytes que contienen la información de la imagen.

Figura 10. Comparación de cadenas hexadecimales de los dos archivos



Fuente: Autores del proyecto.

5 METODOLOGÍA Y MATERIALES

5.1 METODOLOGÍA

Para establecer la metodología a implementar en el desarrollo del proyecto, se tuvo en cuenta la naturaleza del mismo, ya que este se fundamenta en la construcción de un aplicativo de mensajería instantánea para dispositivos móviles, que contenga un módulo de seguridad basado esteganografía, el cual tiene como finalidad que dos personas intercambien información por medio de archivos como imágenes, de esta manera se asegura la confidencialidad de la misma.

Por lo anterior, el desarrollo del software se efectuó con el modelo incremental, una metodología de programación muy utilizada hoy en día, la cual permite que se obtenga un programa final mucho más completo y validando paso a paso. Esta metodología consiste en completar una iteración y/o módulo del software, y hacer pruebas de funcionalidad de lo realizado, luego se vuelve y se hace otra iteración u otro componente y se va probando en cada ciclo finalizado. De esta manera se consigue una evolución constante del desarrollo y permite que se agreguen nuevas especificaciones, opciones y funciones para lograr los objetivos planteados frente al desarrollo del proyecto completo.

Las ventajas de este modelo son:

- Cambiar fácilmente lo planeado, ya que no hace falta que los requerimientos del proyecto estén totalmente definidos desde el comienzo del desarrollo del software.

- Reducir los costos de desarrollo del software, ya que se adapta a las necesidades que surjan.

- Se pueden ir realizando pruebas de cada iteración, lo que permite ir gestionando temas como funcionalidad, desempeño y seguridad.

La principal desventaja es que requiere mayor tiempo desarrollo, ya que en cada iteración se puede cambiar o modificar lo planeado inicialmente.

Para la realización de la aplicación de mensajería instantánea con un módulo de esteganografía y teniendo en cuenta el modelo incremental, se plantearon siete (7) fases:

- Definir los requerimientos del software.

- Establecer las tareas a llevar a cabo en cada iteración para cumplir con los objetivos planeados.

- Diseñar la evolución del software en cada iteración, puesto que cada una de estas tiene una modificación frente a la anterior.

- Desarrollar los incrementos junto con las tareas anteriormente planeadas.

- Validar la funcionalidad de cada iteración. En caso de no conseguir el resultado esperado, se busca la causa de ello y de ser necesario se vuelven a definir las tareas.

- Integración de las iteraciones una vez se validen, para así lograr una evolución del proyecto.

- Efectuar pruebas finales de funcionalidad de la aplicación completa.

5.2 MATERIALES

Para el desarrollo del proyecto, se tuvo en cuenta el trabajo de los dos estudiantes participantes (consultores), así como las herramientas utilizadas: dos equipos de cómputo (con licencias de uso), medios magnéticos, papelería, transporte, servicios de almacenamiento en la nube por hora. Entre estos se encuentran:

- Visual Studio 2015

- SQL Server 2014

- SoapUI de SmartBear[22]

- Servicios Azure[23] de almacenamiento (*storage*), base de datos y aplicación web

- Notepad++[24]

[22] SmartBear. SoapUI. {En línea}. Disponible en https://www.soapui.org/
[23] Microsoft. Azure. {En línea}. Disponible en https://azure.microsoft.com
[24] Notepad++. {En línea}. Disponible en https://notepad-plus-plus.org/

6 DESARROLLO DEL PROYECTO

6.1 CASOS DE USO

De acuerdo a lo descrito anteriormente y teniendo en cuenta las necesidades planteadas en el problema, se definieron los siguientes casos de uso basado en el actor principal de la aplicación, el usuario; el cual es un agente externo que interactúa con esta.

6.1.1 Autenticar usuario. Caso de uso que describe el proceso requerido para la autenticación del usuario en la aplicación, tal como se indica en la Figura 11 y se describe en el Cuadro 2.

Figura 11. Caso de uso para autenticar usuario

Fuente: Autores del proyecto.

Cuadro 2. Descripción del caso de uso de autenticar usuario

Descripción:	El usuario ingresa al aplicativo digitando sus credenciales de autenticación: nombre y contraseña.
Precondiciones:	Estar registrado en el aplicativo
Tipo:	Desarrollo
Flujo básico	
1. El usuario ingresa sus credenciales de ingreso (usuario y contraseña) y da clic en el botón “Aceptar”. 2. La aplicación envía la petición a la capa de acceso o control (*backend*) por medio del servicio web (*web service*). 3. La capa de control recibe la petición y hace la consulta a la base de datos para verificar que el usuario ingresó correctamente sus credenciales de autenticación. 4. La base de datos verifica la petición y responde la consulta a la capa de control. 5. La capa de control procesa la consulta recibida y responde a la aplicación por medio del servicio *web*.	

Cuadro 2. (Continuación)

6. La aplicación recibe la respuesta del servicio *web* y muestra la pantalla principal de la aplicación.

Flujo alterno:

FA1. La aplicación responde que la información digitada por el usuario para ingresar al aplicativo es incorrecta:

1. La aplicación muestra el mensaje en la pantalla “El usuario y/o la contraseña es incorrecta”.
2. El usuario presiona el botón “Aceptar”.
3. El aplicativo borra lo digitado por el usuario.
4. El usuario comienza en el paso número uno (1) del flujo básico.

FA2. La aplicación responde que el usuario digitado por el usuario para ingresar al aplicativo no existe.

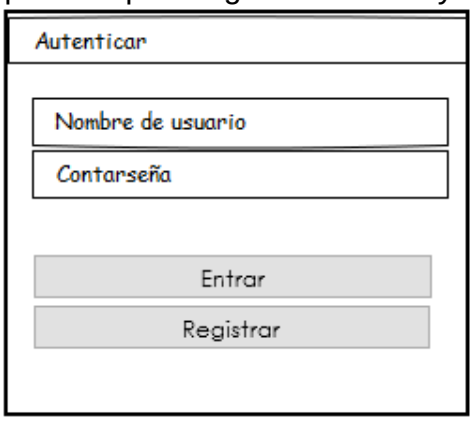
1. La aplicación muestra el mensaje en la pantalla “El usuario no existe, por favor indique una cuenta válida o regístrese”.
2. El usuario presiona el botón “Aceptar”.
3. El aplicativo borra lo digitado por el usuario.

El usuario comienza en el paso número uno (1) del flujo básico o se dirige a la opción de registro.

Requerimientos especiales:

RE1. Una vez el usuario ingresa al aplicativo, se observará la pantalla presentada en la Figura 12 para recibir los datos de la autenticación.

Figura 12. Diseño de la pantalla para ingresar usuario y contraseña

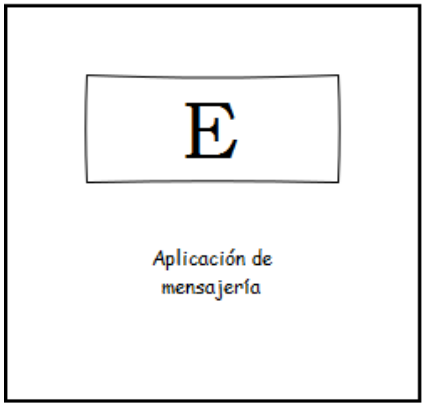


Fuente: Autores del proyecto.

Cuadro 2. (Continuación)

RE2. Al ingresar las credenciales de usuario y contraseña, el usuario observará la pantalla de la Figura 13.

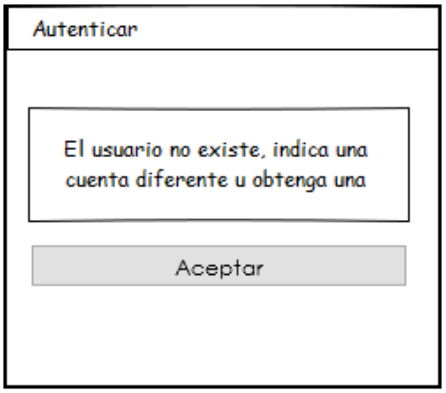
Figura 13. Diseño de la pantalla de inicio del aplicativo



Fuente: Autores del proyecto.

RE3. En los casos que se haya ingresado erróneamente el usuario y/o contraseña, se presentará el mensaje de la Figura 14.

Figura 14. Diseño de la pantalla de autenticación incorrecta

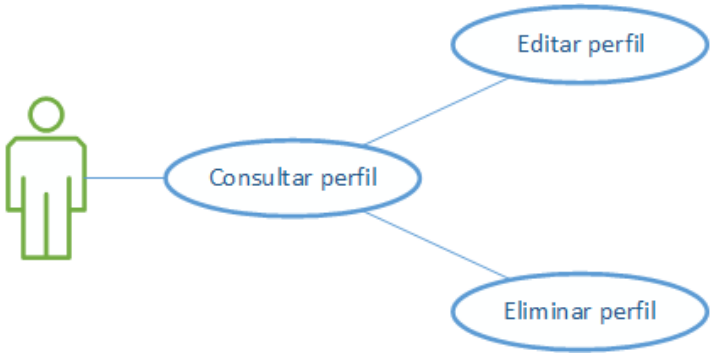


Fuente: Autores del proyecto.

Fuente: Autores del proyecto

6.1.2 Gestión de perfil. Conjunto de casos de uso que describen el proceso de administración de la cuenta de usuario creada en la aplicación, tal como se indica en la Figura 15.

Figura 15. Caso de uso de gestión de perfil



Fuente: Autores del proyecto.

6.1.2.1 Consultar perfil. Caso de uso que describe el proceso de consulta de los datos registrados para el usuario, tal como se presenta en el Cuadro 3.

Cuadro 3. Descripción del caso de uso para consulta del perfil

Descripción:	El usuario va a consultar la información personal registrada previamente en el aplicativo
Precondiciones:	Estar registrado en el aplicativo
Tipo:	Desarrollo
Flujo básico	
1. El usuario ha iniciado sesión digitando su usuario y contraseña. 2. El usuario elige la opción “Perfil”. 3. La aplicación envía la petición a la capa de acceso o control (*backend*) por medio del servicio *web*. 4. La capa de control recibe la petición y hace la consulta a la base de datos. 5. La capa de control procesa la consulta recibida y responde a la aplicación por medio del servicio *web*. 6. La aplicación recibe la respuesta del servicio *web* y muestra en la pantalla de la aplicación las opciones de “Editar Perfil”, “Eliminar Perfil”, la imagen previamente almacenada por el usuario y su nombre y estado.	
Requerimientos especiales:	
RE1. Desde la pantalla principal del aplicativo, el usuario observara la información almacenada previamente en el aplicativo. En la Figura 16 se puede observar la pantalla correspondiente.	

Cuadro 3. (Continuación)

Figura 16. Diseño de la pantalla del perfil del usuario

Fuente: Autores del proyecto.

Fuente: Autores del proyecto.

6.1.2.2 Editar perfil. Caso de uso que describe el proceso a realizar cuando el usuario desea modificar datos de su cuenta de usuario en la aplicación. Su descripción se puede observar en el Cuadro 4.

Cuadro 4. Descripción del caso de uso para editar perfil

Descripción:	El usuario modifica la información personal registrada previamente en el aplicativo
Precondiciones:	Consultar el perfil
Tipo:	Desarrollo

Flujo básico

1. El usuario ha iniciado sesión y se dirige a su pantalla de perfil.
2. El usuario elige la opción denominada “Editar perfil”.
3. La aplicación envía la petición a la capa de acceso o control (*backend*) por medio del servicio *web*.
4. La capa de control recibe la petición y hace la consulta a la base de datos.
5. La capa de control procesa la consulta recibida de la base de datos y responde a la aplicación por medio del servicio *web*.
6. La aplicación recibe la respuesta del servicio *web* y muestra en la pantalla de la aplicación los datos que el usuario ha registrado previamente, como: nombre, imagen del avatar, estado y número de teléfono.

Cuadro 4. (Continuación)

7. El usuario modifica los datos que desea y posteriormente elige la opción denominada “Actualizar perfil”.
8. La aplicación envía la petición a la capa de control por medio del servicio de edición de perfil.
9. La capa de control procesa y valida los datos y, en caso de ser correctos, los actualiza en la base de datos.
10. La capa de control responde al servicio *web* con un mensaje exitoso.
11. La aplicación recibe la respuesta del servicio y muestra al usuario el mensaje “Datos actualizados”.

Flujo alterno:

FA1. La aplicación verifica que la información digitada por el usuario para modificar los datos es incorrecta.

1. La capa de control procesa y valida los datos y responde con fallo en la actualización (Esto puede ser porque el usuario escribió sólo números en su “nombre”, letras en el campo de “teléfono”, subió archivos de imagen no aceptados o la casilla de contraseña estaba vacía).
2. La recibe la respuesta a través del servicio *web* y muestra el mensaje en la pantalla “Los datos ingresados son inválidos”.
3. El usuario presiona el botón “Aceptar”.
4. El aplicativo borra lo digitado por el usuario.
5. El usuario inicia en el paso número siete (7) del flujo básico.

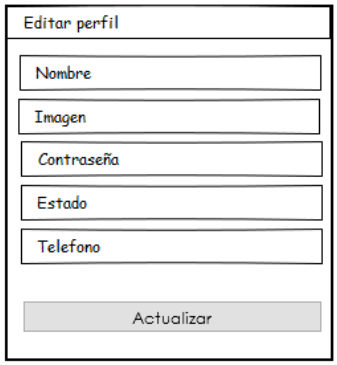
FA2. El usuario no modifica la información del perfil.

1. En el punto seis (6) del flujo básico, el usuario no modifica ninguno de los datos que ha registrado previamente y da clic en otra pestaña del aplicativo.
2. La aplicación recibe la respuesta y muestra en pantalla los datos de perfil.

Requerimientos especiales:

RE1. Desde la opción de perfil, el usuario podrá editar la información almacenada previamente en el aplicativo, tal como se presenta en la Figura 17.

Figura 17. Diseño de la pantalla de la información almacenada en el aplicativo



Fuente: Autores del proyecto.

Cuadro 4. (Continuación)

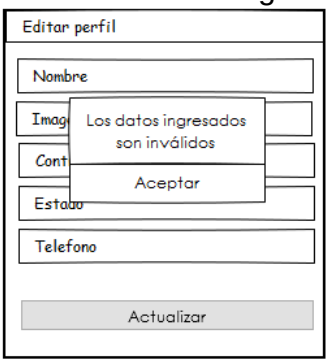
RE2. Una vez el usuario ha actualizado sus datos en la opción de perfil, se observará la pantalla de la Figura 18.

Figura 18. Diseño de la pantalla de los datos actualizados exitosamente

Fuente: Autores del proyecto.

RE3. Cuando el usuario haya ingresado algún dato del perfil incorrecto, se presentará el mensaje de la Figura 19.

Figura 19. Diseño de la pantalla de los datos ingresados son incorrecto



Fuente: Autores del proyecto.

Fuente: Autores del proyecto.

6.1.2.3 Eliminar perfil. Caso de uso que presenta los pasos a seguir para la eliminación de una cuenta de usuario. Su descripción se puede observar en el Cuadro 5.

Cuadro 5. Descripción del caso de uso para eliminar perfil

Descripción:	El usuario desea eliminar su cuenta de usuario del aplicativo
Precondiciones:	Consultar el perfil
Tipo:	Desarrollo

Cuadro 5. (Continuación)

Flujo básico

1. El usuario selecciona el botón de “Eliminar” de la pantalla de perfil del aplicativo.
2. El aplicativo muestra el mensaje “¿Está seguro que desea eliminar la cuenta?”, y el usuario hace clic en el botón “Aceptar”.
3. El aplicativo recibe la confirmación y la envía por medio del servicio *web* a la capa de control o acceso (*backend*).
4. La capa de control recibe la solicitud y envía la instrucción de borrado a la base datos, la cual la procesa y responde.
5. La capa de control responde a la aplicación por medio del servicio *web*.
6. La aplicación recibe la respuesta y muestra en pantalla “Cuenta eliminada”, el usuario hace clic en “Aceptar”.
7. La aplicación cierra la sesión y muestra nuevamente la pantalla de autenticación de usuario.

Flujo alterno:

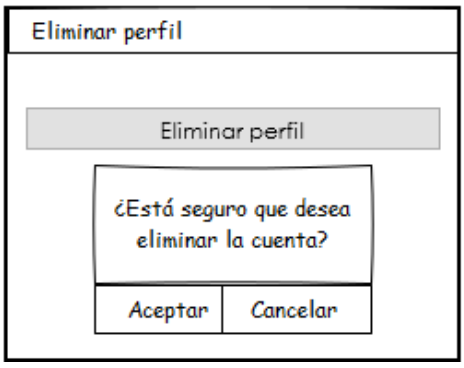
FA1. El usuario selecciona la opción de no eliminar el perfil.

1. En el punto dos (2) del flujo básico, el usuario selecciona la opción denominada “Cancelar”.
2. El aplicativo recibe la respuesta y muestra en pantalla el perfil de inicio del usuario.

Requerimientos especiales:

RE1. Desde la pantalla de perfil, el usuario podrá eliminar su cuenta. Al dar clic en esta opción, se presentará el mensaje de la Figura 20.

Figura 20. Diseño de la pantalla de eliminar cuenta

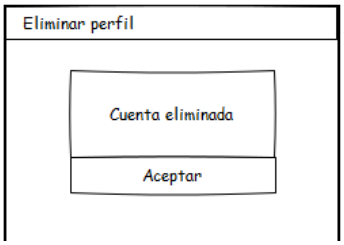


Fuente: Autores del proyecto.

Cuadro 5. (Continuación)

RE2. Al hacer clic en la opción de eliminar cuenta y confirmar la intención de borrarla, el usuario observará la pantalla de la Figura 21.

Figura 21. Diseño de la pantalla de cuenta eliminada

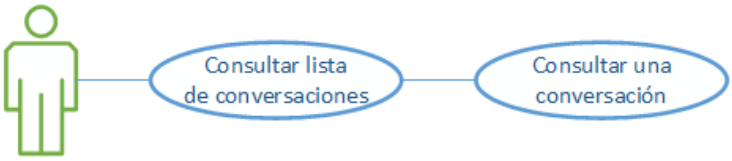


Fuente: Autores del proyecto.

Fuente: Autores del proyecto.

6.1.3 Conversaciones. Conjunto de casos de uso que contemplan las tareas de gestión, creación y consulta de conversaciones dentro de la aplicación, tal como se puede observar en la Figura 22.

Figura 22. Caso de uso para la consulta de conversaciones



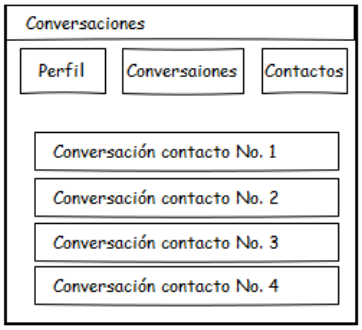
Fuente: Autores del proyecto.

6.1.3.1 Consultar lista de conversaciones. Caso de uso para observar en pantalla la lista de conversaciones que tenga el usuario. Sus pasos se describen en el Cuadro 6.

Cuadro 6. Descripción del caso de uso para consultar conversaciones

Descripción:	El usuario desea consultar la lista de las conversaciones llevadas a cabo con los contactos que tiene en la aplicación
Precondiciones:	Ingresar al sistema (autenticación)
Tipo:	Desarrollo
Flujo básico	
1. El usuario da clic en la opción “Conversaciones”. 2. La aplicación envía la petición por medio del servicio *web* a la capa de control. 3. La capa de control recibe la petición y hace la consulta a la base de datos.	

Cuadro 6. (Continuación)

4. La base de datos responde a la capa de control, la cual procesa la información recibida y responde a través del servicio *web*. 5. La aplicación recibe la respuesta del servicio *web* y presenta al usuario la lista de conversaciones según la información recibida.
Flujo alterno
FA1. El usuario no ha establecido conversaciones con algún contacto. 1. En el punto cinco (5) del flujo básico, la aplicación no presenta ninguna conversación, ya que el usuario no ha establecido alguna comunicación por medio de esta. 2. El aplicativo luego de unos segundos, presenta la lista de contactos para que el usuario pueda definir con quien efectuar una conversación.
Requerimientos especiales
RE1. Al dar clic en la opción de conversaciones, el usuario podrá observar la lista de las conversaciones llevadas a cabo, tal como se indica en la Figura 23. Figura 23. Diseño de la pantalla de la lista de conversaciones    Fuente: Autores del proyecto.

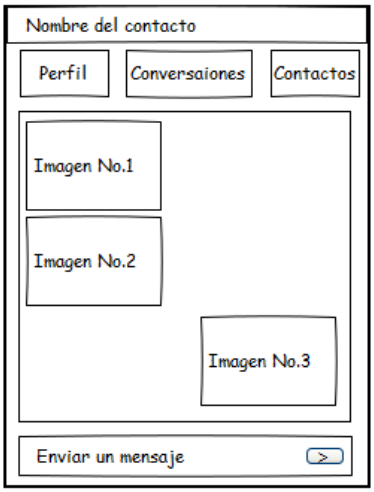
Fuente: Autores del proyecto.

6.1.3.2 Consultar una conversación. Caso de uso que describe el procedimiento a seguir por parte de la aplicación para presentar al usuario una conversación que haya seleccionado previamente. Se presenta un resumen de cada uno de sus pasos en el Cuadro 7.

Cuadro 7. Descripción del caso de uso para consultar una conversación

Descripción:	El usuario desea observar una de las conversaciones que mantiene con un contacto
Precondiciones:	Consultar la lista de conversaciones o contactos
Tipo:	Desarrollo

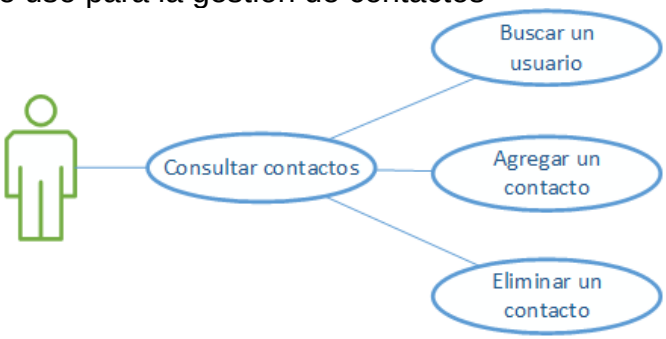
Cuadro 7. (Continuación)

Flujo básico
1. El usuario da clic en la conversación con el contacto definido. 2. La aplicación envía la petición por medio del servicio *web* a la capa de control. 3. La capa de control recibe la solicitud y hace la consulta a la base de datos. 4. La base de datos responde a la consulta realizada y envía la información resultante a la capa de control. 5. La capa de control procesa la información recibida y responde al servicio *web.* 6. La aplicación recibe la respuesta a través del servicio *web* y presenta la lista de mensajes al usuario.
Requerimientos especiales
RE1. Cuando el usuario de clic a la conversación con el contacto definido, podrá observar los mensajes intercambiados entre estos, tal como se muestra en la Figura 24. Figura 24. Diseño de la pantalla de la lista de conversaciones punto a punto    Fuente: Autores del proyecto.

Fuente: Autores del proyecto.

6.1.4 Gestión de contactos. Conjunto de casos de uso para las tareas de gestión de contactos como consulta, creación, etc. Cada una de estas tareas se presentan en la Figura 25.

Figura 25. Caso de uso para la gestión de contactos



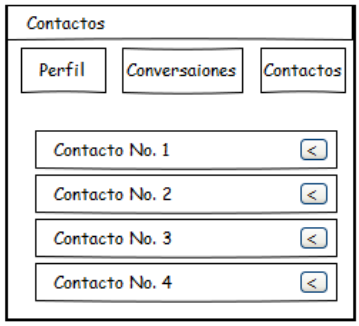
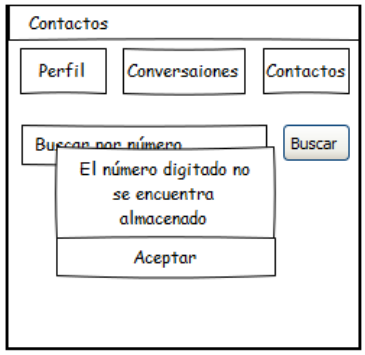
Fuente: Autores del proyecto.

6.1.4.1 Consultar contactos. Caso de uso que presenta los pasos a seguir para consultar los contactos del usuario en la aplicación. Su descripción se puede observar en el Cuadro 8.

Cuadro 8. Descripción del caso de uso para consultar de contactos

Descripción:	El usuario desea consultar la lista de contactos que tiene en la aplicación
Precondiciones:	Ingresar al sistema (autenticación)
Tipo:	Desarrollo
Flujo básico	
1. El usuario da clic en la opción “Contactos”. 2. La aplicación envía la petición por medio del servicio *web* a la capa de control. 3. La capa de control recibe la petición y hace la consulta a la base de datos. 4. La base de datos responde a la capa de control, la cual procesa la información recibida y responde a través del servicio *web*. 5. La aplicación recibe la respuesta del servicio *web* y presenta al usuario la lista de contactos que tiene en el aplicativo según la información recibida.	
Flujo alterno:	
FA1. El usuario no tiene contactos almacenados en la aplicación. 1. En el punto cinco (5) del flujo básico, la aplicación no presenta ningún contacto, ya que el usuario no ha agregado algún contacto a su lista. 2. En caso de que el usuario desee agregar un contacto, lo podrá hacer buscándolo por medio de su número telefónico.	
Requerimientos especiales:	
RE1. Al darle clic en contactos, el usuario observará la lista de los contactos que ha agregado en el aplicativo, y que se puede revisar en la Figura 26.	

Cuadro 8. (Continuación)

Requerimientos especiales:
Figura 26. Diseño de la pantalla de la lista de contactos    Fuente: Autores del proyecto. **RE2.** Una vez el usuario ha digitado el número del contacto a buscar, que no se encuentra almacenado en el aplicativo, se observará el mensaje de la Figura 27. Figura 27. Diseño de la pantalla de contacto no almacenado    Fuente: Autores del proyecto.

Fuente: Autores del proyecto.

6.1.4.2 Buscar un usuario. Caso de uso que describe el procedimiento de búsqueda de un usuario al interior de la aplicación. Descrito en el Cuadro 9.

Cuadro 9. Descripción del caso de uso para la búsqueda de un contacto

Descripción:	El usuario desea buscar un contacto en la lista sus contactos en el aplicativo
Precondiciones:	Consultar la lista de contactos
Tipo:	Desarrollo

Cuadro 9. (Continuación)

Flujo básico
1. El usuario digita el número del contacto y le da clic en la opción “Buscar”. 2. La aplicación envía la petición por medio del servicio *web* a la capa de control o acceso (*backend*). 3. La capa de control recibe la solicitud y hace la consulta a la base de datos. 4. La base de datos responde a la consulta realizada y envía la información resultante a la capa de control. 5. La capa de control procesa la información recibida y responde al servicio *web.* 6. La aplicación recibe la respuesta a través del servicio *web* y presenta el contacto almacenado con el número digitado.
Flujo alterno:
FA1. El usuario no tiene almacenado el contacto con el número digitado en el aplicativo. 1. En el punto seis (6) del flujo básico, la aplicación muestra el mensaje “El número digitado no se encuentra almacenado”, y el usuario da clic en el botón “Aceptar”. 2. El aplicativo muestra la lista de contactos almacenados.
Requerimientos especiales:
RE1. Cuando el usuario desea buscar algún contacto en la lista de sus contactos al interior de la aplicación, observará la pantalla de la Figura 28. Figura 28. Diseño de la pantalla del contacto buscado    Fuente: Autores del proyecto.

Fuente: Autores del proyecto.

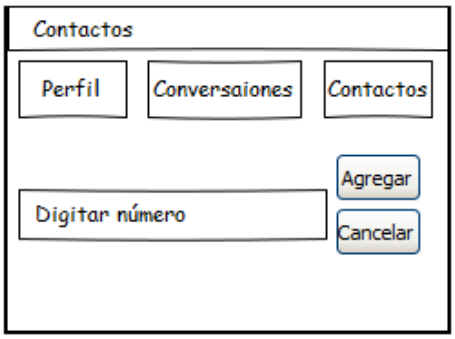
6.1.4.3 Agregar un contacto. Caso de uso que describe el procedimiento que debe seguir el usuario para agregar un contacto a su lista de contactos. Esto se describe en el Cuadro 10.

Cuadro 10. Descripción del caso de uso para crear (agregar) un contacto

Descripción:	El usuario desea crear un contacto en la lista
Precondiciones:	Consultar la lista de contactos
Tipo:	Desarrollo
Flujo básico	
1. El usuario hace clic en el botón “Agregar”. 2. El aplicativo recibe la petición y la envía por medio del servicio *web* a la capa de control (*backend*). 3. La capa de control recibe la solicitud y envía la instrucción correspondiente a la base datos. 4. La base de datos crea el registro correspondiente en el usuario y responde a la capa de control. 5. La capa de control procesa la información recibida y se la envía a la aplicación por medio del servicio *web*. 6. La aplicación recibe la respuesta y muestra en pantalla una celda vacía, para que el usuario ingrese el número telefónico del nuevo contacto. 7. El usuario digita la información correspondiente al número telefónico del nuevo contacto y le da clic en “Agregar”. 8. El aplicativo recibe la petición y la envía por medio del servicio *web* a la capa de control (*backend*). 9. La capa de control recibe la solicitud y envía la instrucción correspondiente. 10. La base de datos actualiza el registro del nuevo contacto del usuario y responde a la capa de control. 11. La capa de control procesa la información recibida y se la envía a la aplicación por medio del servicio *web*. 12. La aplicación recibe la respuesta y muestra en pantalla el mensaje “Contacto creado” y el usuario da clic en “Aceptar”.	
Flujo alterno:	
FA1. El usuario selecciona la opción de no crear el contacto. 1. En el punto siete (7) del flujo básico el usuario selecciona la opción “Cancelar” 2. La aplicación muestra la lista de todos los contactos.	
Requerimientos especiales:	
RE1. En la pantalla presentada en la Figura 29, se observa cómo se puede visualizar la opción de agregar un contacto en el aplicativo.	

Cuadro 10. (Continuación)

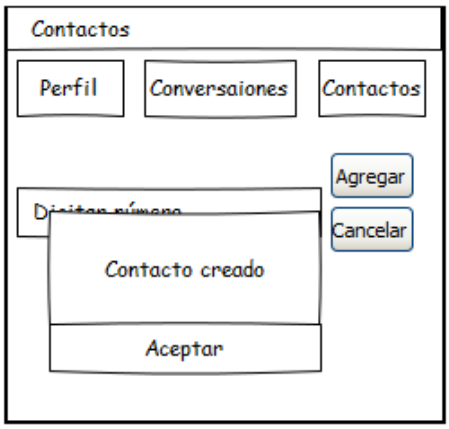
Figura 29. Diseño de la pantalla con la opción de agregar contacto.



Fuente: Autores del proyecto.

RE2. Una vez el usuario ha creado el contacto en el aplicativo, se presentará el mensaje de la Figura 30.

Figura 30. Diseño de la pantalla de contacto creado



Fuente: Autores del proyecto.

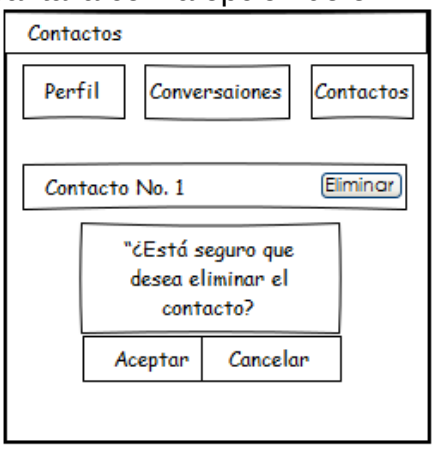
Fuente: Autores del proyecto.

6.1.4.4 Eliminar un contacto. Caso de uso con la descripción del procedimiento para el proceso de eliminar un contacto de la lista de contactos del usuario. Su descripción se puede observar en el Cuadro 11.

Cuadro 11. Descripción del caso de uso para eliminar un contacto

Descripción:	El usuario desea eliminar un contacto de la lista
Precondiciones:	Consultar la lista de contactos
Tipo:	Desarrollo

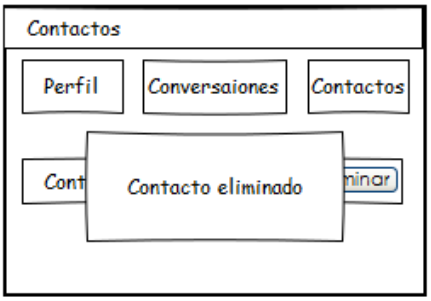
Cuadro 11. (Continuación)

Flujo básico
1. El usuario selecciona el contacto que desea eliminar y hace clic en el botón “Eliminar”. 2. El aplicativo presenta el mensaje “¿Está seguro que desea eliminar el contacto?”, y el usuario da clic en el botón de aceptar. 3. El aplicativo recibe la petición y la envía por medio del servicio web a la capa de control (*backend*). 4. La capa de control recibe la solicitud y envía la instrucción correspondiente a la base datos. 5. La base de datos actualiza el registro correspondiente señalando al usuario como eliminado y responde a la capa de control. 6. La capa de control procesa la información recibida y se la envía a la aplicación por medio del servicio *web*. 7. La aplicación recibe la respuesta y muestra en pantalla el mensaje “Contacto eliminado”.
Flujo alterno:
FA1. El usuario selecciona la opción de no eliminar el contacto. 1. En el punto dos (2) del flujo básico el usuario selecciona la opción “Cancelar” 2. La aplicación muestra la lista de todos los contactos.
Requerimientos especiales:
RE1. Cuando el usuario desea eliminar algún contacto de su lista, se presentará el mensaje indicado en la Figura 31. Figura 31. Diseño de la pantalla con la opción de eliminar contacto    Fuente: Autores del proyecto.

Cuadro 11. (Continuación)

RE2. Una vez el usuario decide eliminar por completo el contacto de su lista, se enseñará la pantalla con el mensaje de “Contacto eliminado”, tal como lo muestra la Figura 32.

Figura 32. Diseño de la pantalla de contacto eliminado



Fuente: Autores del proyecto.

Fuente: Autores del proyecto.

6.2 REQUERIMIENTOS DE SEGURIDAD

Para el desarrollo de la aplicación de mensajería instantánea se tuvieron en cuenta los siguientes requerimientos de seguridad de la información:

6.2.1 Requerimientos del proyecto. Los siguientes requerimientos de seguridad hacen parte del alcance del proyecto, puesto que tratan temas como la esteganografía, el cifrado y la confidencialidad en los mensajes. Por lo tanto, deben tenerse en cuenta y ser evaluados para considerar el éxito del mismo:

6.2.1.1 Lectura de mensajes. Los mensajes que se encuentran ocultos en una imagen, podrán ser visualizados por el usuario si digita una clave. De esta manera se asegura la confidencialidad de la información en caso que un intruso pueda tener acceso al dispositivo móvil.

6.2.1.2 Almacenamiento de mensajes. Los mensajes no deben almacenarse en la base de datos, de tal forma que si un intruso accede a esta, no pueda observar el flujo de información de cualquier conversación.

6.2.1.3 Mensajes ocultos y cifrados. Los mensajes que se almacenen dentro de la aplicación deben estar cifrados (criptografía) y ocultos en una imagen

(esteganografía). Lo anterior con el fin de aumentar la complejidad de lectura del mensaje en caso de ataques o intrusiones.

6.2.1.4 Diferencia entre contraseñas. La clave para descifrar los mensajes debe ser diferente a la clave de autenticación, para evitar que al conocer o descubrir una de ellas, un tercero pueda acceder directamente a los mensajes.

6.2.1.5 Almacenamiento de contraseñas. Tanto la clave de autenticación como la de lectura del mensaje, no deben ser almacenadas en la base de datos de forma plana, se debe contar con un sistema que no permita a intrusos conocerla fácilmente, en caso de que logren acceder a la capa de datos.

6.2.2 Requerimientos adicionales. Adicional a los requerimientos anteriores, se recomienda la aplicación de los siguientes requisitos, pensando en un proyecto funcional en ambiente productivo:

6.2.2.1 Comunicaciones seguras. Elementos como mensajes de una conversación y contraseñas no deben viajar en formato plano en la comunicación entre servidor y aplicación y viceversa. Estos deben transmitirse a través de un canal cifrado, esto para asegurar la integridad y confidencialidad de la información frente a un ataque o intercepción del mismo.

6.2.2.2 Tiempo de lectura del mensaje. Para proteger la confidencialidad de los mensajes visualizados, una vez el usuario ingrese la clave de lectura del mensaje, el aplicativo deberá presentar durante un tiempo determinado parametrizable en base de datos y posteriormente cerrar la ventana de visualización.

6.2.2.3 Servidores independientes. El aplicativo debe contar con servidores independientes para las bases de datos, para el repositorio o banco de imágenes y para la aplicación de control. De esta manera, si uno de los ambientes se ve afectado por una intrusión, los otros no se verán comprometidos.

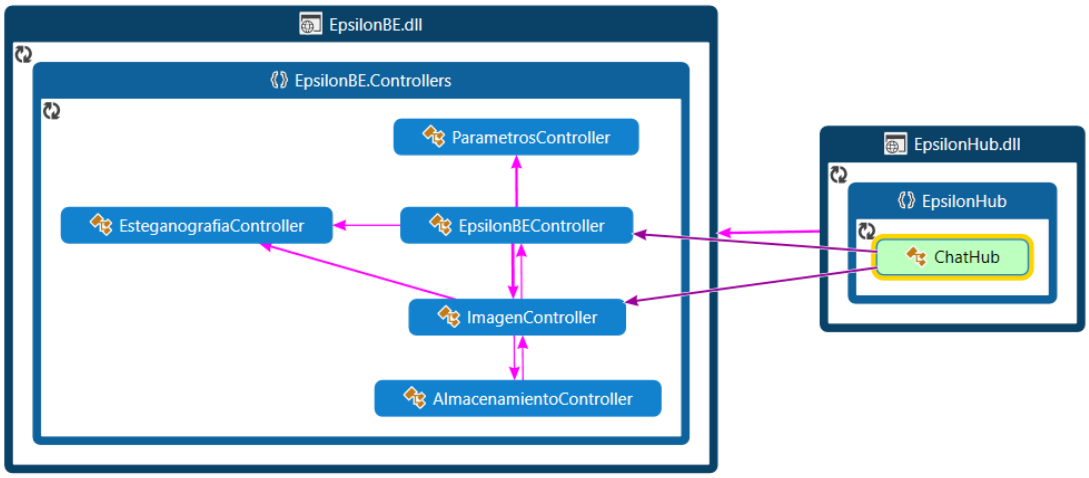
6.2.2.4 Usuario único. El aplicativo debe contar con una validación inicial de creación del usuario, que deberá registrar el número telefónico y este será verificado a través de un mensaje de texto con un código de confirmación. Esto con el fin de evitar una posible suplantación de identidad.

6.2.2.5 Sesión única. Un usuario sólo podrá tener una sesión activa a la vez en el aplicativo. Si un mismo usuario intenta acceder desde dos dispositivos o instancias diferentes del sistema, el aplicativo no lo deberá permitir.

6.3 DIAGRAMAS DE CLASES

La Figura 33 presenta las relaciones entre las principales clases que componen la aplicación.

Figura 33. Relaciones entre las clases principales

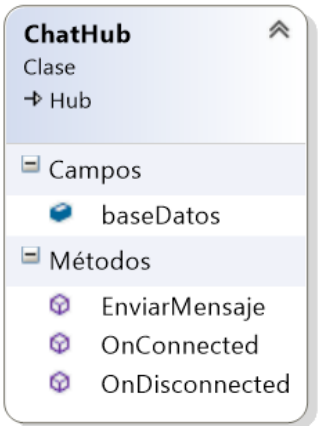


Fuente: Autores del proyecto

Las clases que hacen parte de la aplicación son:

6.3.1 Componente de comunicación: ChatHub. Recibe solicitudes de conexión, desconexión y peticiones para envío de mensajes. Tiene conexión con la base de datos con el fin de consultar o escribir directamente sobre la tabla de conexiones de la misma. En la Figura 34, se puede observar su estructura.

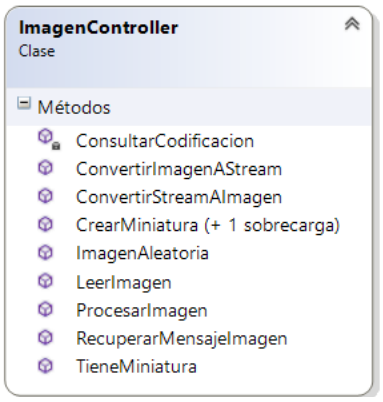
Figura 34. Estructura de la clase ChatHub



Fuente: Autores del proyecto (generado con Visual Studio desde el código).

6.3.2 Manejo de imágenes: ImagenController. Gestiona las tareas relacionadas con el manejo de la imagen, entre ellas las conversiones de imagen a *stream* y viceversa, como pasos inicial y final del procedimiento de criptografía. También es el componente encargado de comunicarse con el repositorio de imágenes (EpsilonBlob). La Figura 35 presenta los métodos que la conforman.

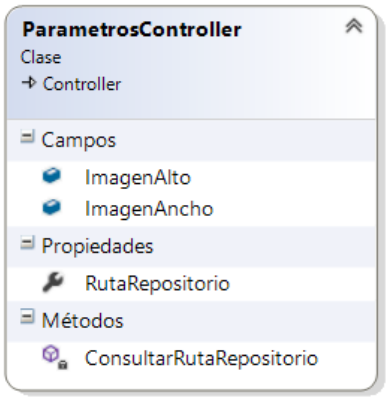
Figura 35. Estructura de la clase ImagenController



Fuente: Autores del proyecto (generado con Visual Studio desde el código).

6.3.3 Manejo de parámetros: ParametrosController. Es la encargada de entregar los parámetros a nivel del *backend* de la aplicación, como la ruta del repositorio o las dimensiones de la imagen. En la Figura 36 se presentan sus campos, propiedades y métodos.

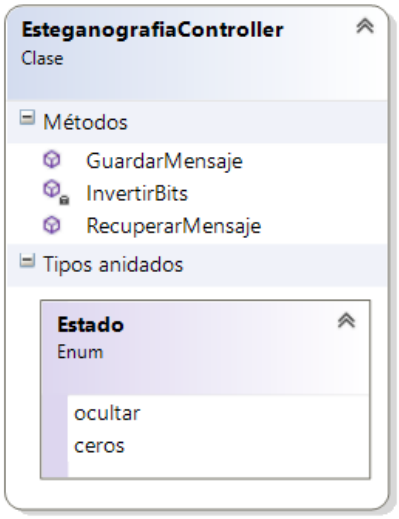
Figura 36. Estructura de la clase ParametrosController



Fuente: Autores del proyecto (generado con Visual Studio desde el código).

6.3.4 Componente de esteganografía: EsteganografiaController. Contiene los métodos encargados de ocultar el mensaje en la imagen, tanto para guardar el mensaje como para recuperarlo. La Figura 37 señala las partes que la componen.

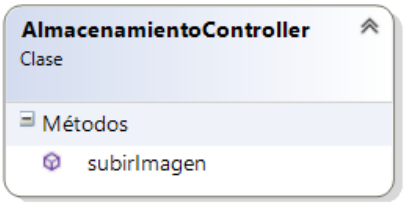
Figura 37. Estructura de la clase *EsteganografiaController*



Fuente: Autores del proyecto (generado con Visual Studio desde el código).

6.3.5 Gestión del repositorio de imágenes: AlmacenamientoController. Componente encargado de guardar la imagen en el repositorio. En la Figura 38 se puede ver su estructura.

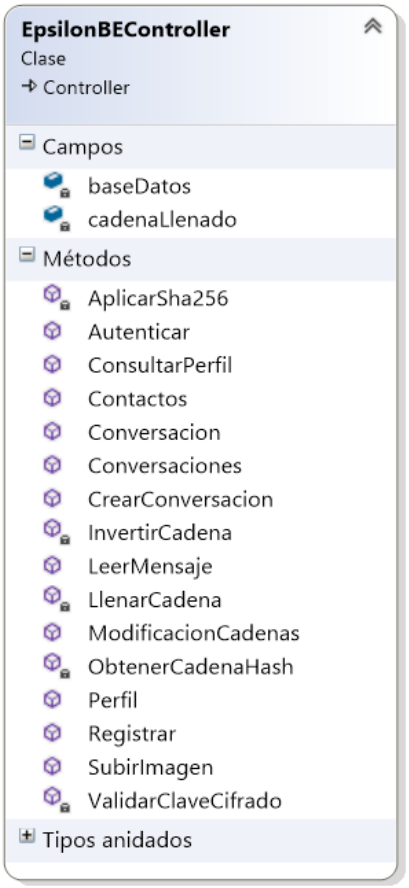
Figura 38. Estructura de la clase *AlmacenamientoController*



Fuente: Autores del proyecto (generado con Visual Studio desde el código).

6.3.6 Servicios y manejo de la aplicación: EpsilonBEController. Es el componente principal o núcleo del sistema, que recibe las solicitudes de la aplicación y las distribuye entre las demás clases o se comunica con la base de datos, según sea el caso. En la Figura 39 se observa su composición.

Figura 39. Estructura de la clase *EpsilonBEController*

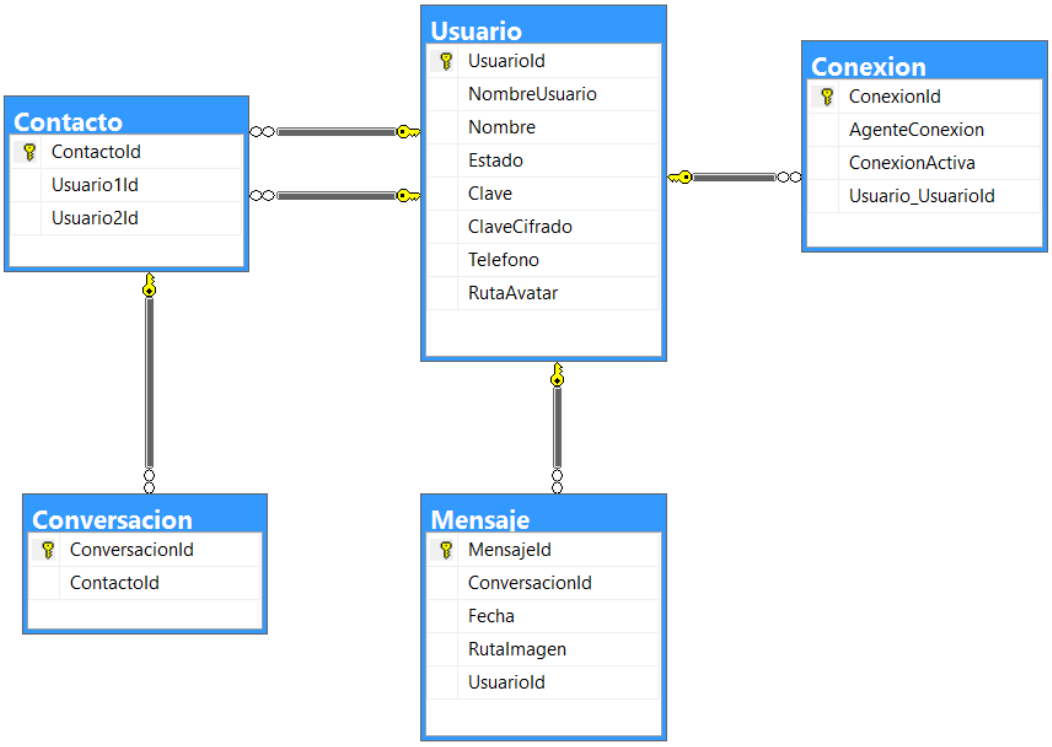


Fuente: Autores del proyecto (generado con Visual Studio desde el código).

6.4 DIAGRAMA DE BASES DE DATOS

En la Figura 40 se presenta el diagrama de las entidades y relaciones de la capa de datos de la aplicación, junto con sus propiedades:

Figura 40. Diagrama de base de datos



Fuente: Autores del proyecto (generado desde SQL Server Management Studio).

6.5 DISEÑO DE LA APLICACIÓN

La aplicación está compuesta por cinco componentes principales:

6.5.1 Base de datos (EpsilonBD). Es una base de datos en SQL Server que se encarga de almacenar toda la información relacionada con la aplicación: usuarios, contactos, conversaciones (con las rutas de las imágenes que contienen los mensajes) y conexiones:

6.5.1.1 Tabla de usuarios. Esta contiene los datos de cada usuario registrado en la aplicación. En la Figura 41 se presentan las columnas que lo conforman.

• UsuarioId: código interno del usuario, es la llave primaria de la tabla.

• NombreUsuario: cadena de caracteres que debe definir el usuario al momento del registro y que debe ser único, ya que será el identificador del mismo en la aplicación.

• Nombre: nombre completo del usuario.

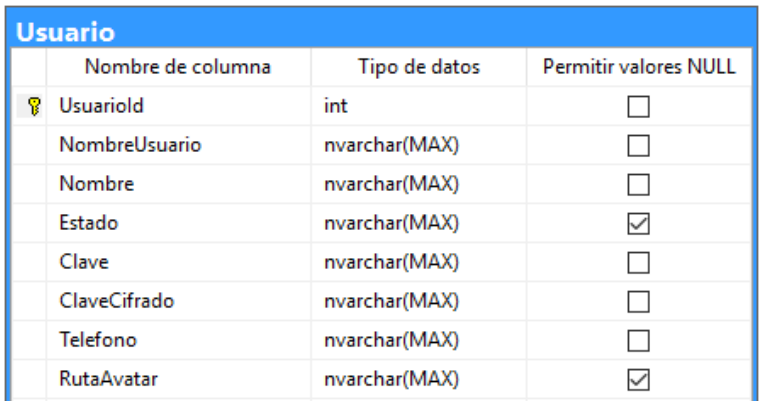
• Estado: etiqueta o mensaje que el usuario quiera dejar como frase de perfil, la cual verán los demás contactos. Esta puede ser editada ocasionalmente.

• Clave: contraseña de ingreso al sistema almacenada como hash (SHA-256) para ser verificada en el momento del ingreso al sistema.

• Telefono: campo numérico en el que se almacena el número de teléfono del usuario. Debe ser único, sirve como llave de búsqueda para nuevos contactos y se propone como alternativa de verificación (envío de mensaje de texto) al momento del registro.

• RutaAvatar: campo de texto donde se almacena el nombre del archivo en el repositorio de imágenes para el directorio de perfiles. Esta ruta dirige a la imagen de identificación del usuario y es visible para todos sus contactos.

Figura 41. Tabla de usuarios en la base de datos



Fuente: Autores del proyecto (desde SQL Server Management Studio).

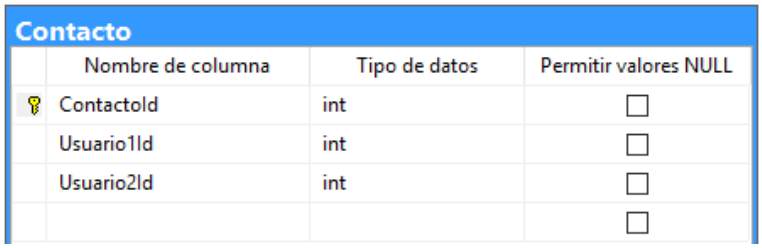
6.5.1.2 Tabla de contactos. Contiene las conexiones creadas entre los diferentes usuarios, luego que alguno de ellos envíe una solicitud y esta sea aceptada. Los campos que la conforman se presentan en la Figura 42.

• ContactoId: código interno de la relación creada entre usuarios.

• Usuario1Id: código del primer usuario que conforma la relación de contacto.

• Usuario2Id: código del segundo usuario que conforma la relación de contacto.

Figura 42. Tabla de contactos en la base de datos



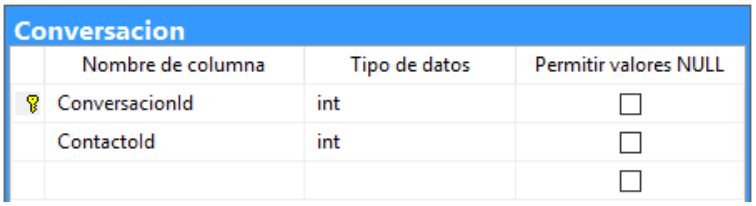
Fuente: Autores del proyecto (desde SQL Server Management Studio).

6.5.1.3 Tabla de conversaciones. Contiene los registros de las conversaciones creadas entre contactos. Está creada de forma independiente a la tabla de contactos para un futuro soporte de conversaciones entre más de dos usuarios (grupos). Su estructura se observa en la Figura 43. Los campos que la conforman son:

• ConversacionId: código interno del registro de la conversación.

• ContactoId: código del registro correspondiente en la tabla de contactos.

Figura 43. Tabla de conversaciones en la base de datos



Fuente: Autores del proyecto (desde SQL Server Management Studio).

6.5.1.4 Tabla de mensajes. Contiene los registros de cada mensaje que se envía a través de la aplicación. Sin embargo, esta tabla no almacena los mensajes como tal, sino la ruta de la imagen en la que fueron incrustados. En la Figura 44 se observa su estructura. Los campos que la conforman son:

• MensajeId: código interno del registro del mensaje.

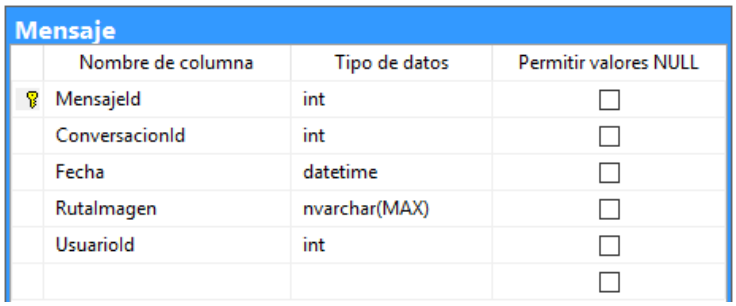
• ConversacionId: código del registro de la conversación a la que pertenece el mensaje.

• Fecha: momento en el que fue enviado el mensaje.

• RutaImagen: ubicación en el repositorio (EpsilonBlob) de la imagen que contiene el mensaje mediante esteganografía.

•UsuarioId: código del autor del mensaje.

Figura 44. Tabla de mensajes en la base de datos



Fuente: Autores del proyecto (desde SQL Server Management Studio).

6.5.1.5 Tabla de conexiones. Almacena los registros de conexiones existentes en el momento para la aplicación. Cuando un usuario ingresa, se crea un nuevo registro con el valor booleano *ConexionActiva* en valor verdadero. Una vez el usuario cierra sesión o al pasar el tiempo máximo permitido (*timeout*), este valor pasa a ser falso. Lo que obliga a que se cree una nueva a la siguiente ocasión de ingreso del usuario. Los campos que la conforman y que se muestran en la Figura 45, son:

• ConexionId: código interno del registro del registro de la conexión.

• AgenteConexion: características del equipo desde el que se conecta el usuario.

• ConexionActiva: campo *booleano* que indica si el usuario se encuentra conectado o no.

• UsuarioId: código del registro del usuario conectado.

Figura 45. Tabla de conexiones en la base de datos

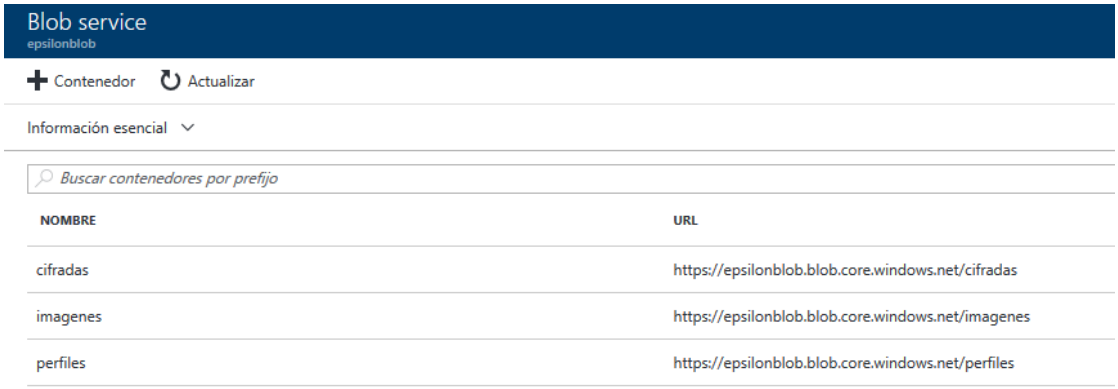
Conexion

	Nombre de columna	Tipo de datos	Permitir valores NULL
🔑	ConexionId	nvarchar(128)	☐
	AgenteConexion	nvarchar(MAX)	☐
	ConexionActiva	bit	☐
	Usuario_UsuarioId	int	☑
			☐

Fuente: Autores del proyecto.

6.5.2 Repositorio o banco de imágenes (EpsilonBlob). Es un servidor encargado de almacenar las imágenes como tipo blob, el cual está dividido en tres directorios: perfiles, originales y cifradas, tal como lo muestra la Figura 46.

Figura 46. Directorios del repositorio de imágenes



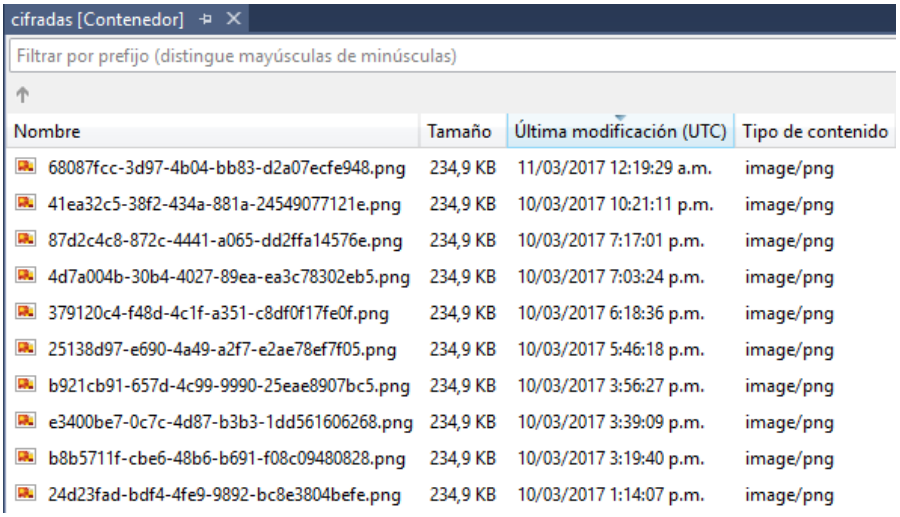
Fuente: Autores del proyecto.

• Cifradas: directorio que almacena las imágenes que contienen cada mensaje enviado. Se crean en el momento en el que un usuario envía un mensaje determinado y se compone de una imagen original almacenada en el directorio imágenes y del mensaje incrustado con técnicas de esteganografía. El nombre de cada archivo se asigna mediante un identificador único global (*GUID*), tal como se observa en la Figura 47.

• Imágenes: directorio que contiene las imágenes originales que el administrador carga y que funcionan como archivos base para incrustar los mensajes enviados. Pueden agregarse más o modificarse periódicamente. La aplicación utiliza un método para seleccionar aleatoriamente una de las alojadas en este repositorio.

• Perfiles: almacena las imágenes de los perfiles de cada usuario (avatares).

Figura 47. Contenido del directorio de imágenes cifradas



Fuente: Autores del proyecto.

6.5.3 Aplicación de servidor o backend (EpsilonBE). Encargada de controlar todas las peticiones de la aplicación móvil diferentes al tema de comunicación e integrar la comunicación entre la capa de la aplicación móvil y las de información: repositorio de imágenes y base de datos. En la Figura 48 se presenta el mapa de código de este componente.

Contiene los siguientes módulos:

- Usuarios: Registrar, Perfil, ConsultarPerfil, DatosUsuario

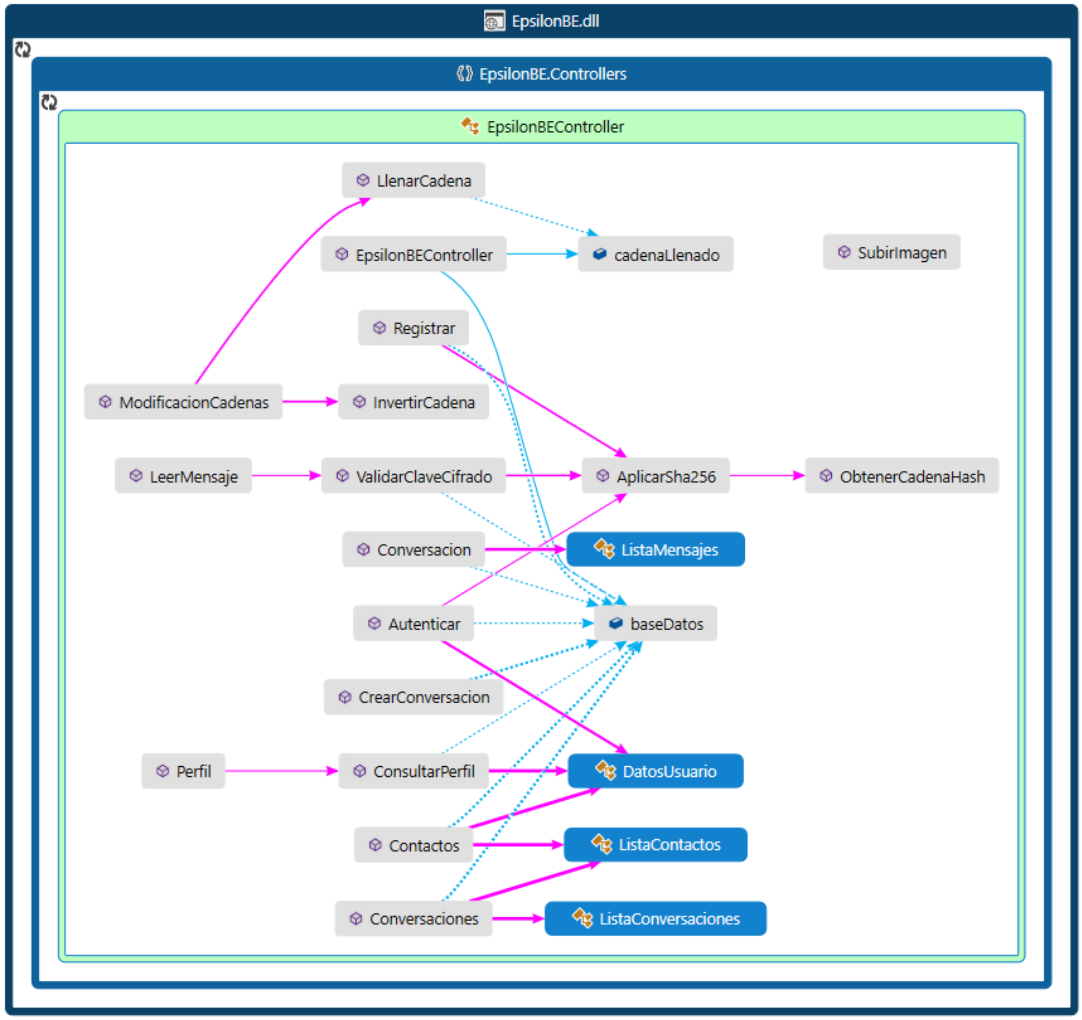
- Contactos: Contactos, ListaContactos

- Gestión de conversaciones: CrearConversacion, ListaConversaciones, Conversacion

- Manejo de imagen: SubirImagen

- Esteganografía: ModificacionCadenas, LlenarCadena, ObtenerCadenaHash, CadenaLlenado, InvertirCadenas, ValidarClaveCifrado, AplicarSha256

Figura 48. Mapa de código de EpsilonBE



Fuente: Autores del proyecto.

6.5.3.1 Clases del controlador. Se presentan en la Figura 49. Las siguientes son las clases principales del controlador de *backend*:

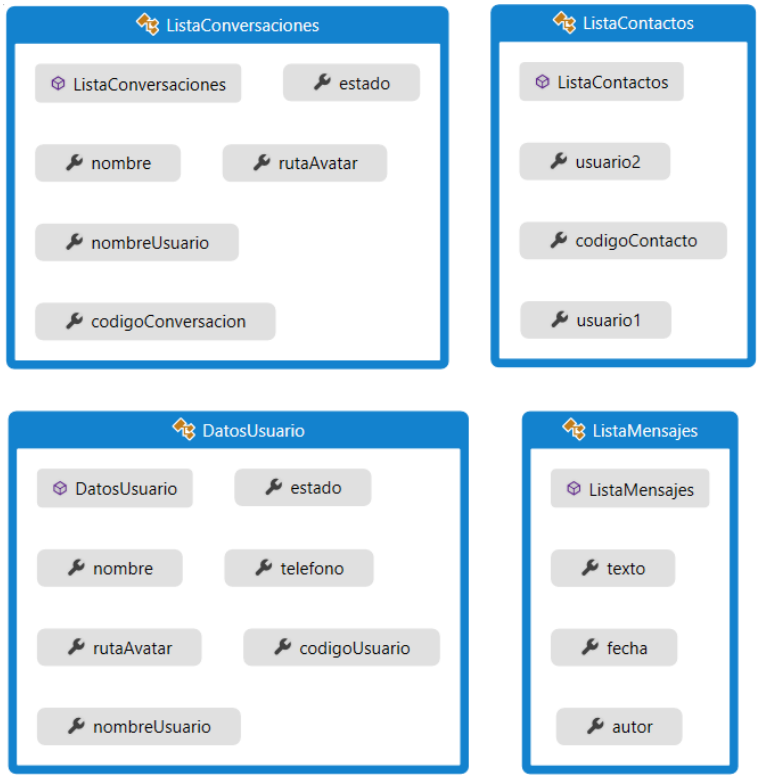
• *ListaConversaciones*: clase creada para responder a la consulta del servicio de conversaciones.

• *ListaMensajes*: con esta estructura se responde a la consulta del servicio de mensajes por parte de la aplicación.

• *ListaContactos*: con esta estructura se responde a la consulta del servicio de conversaciones.

• *DatosUsuario*: clase creada para responder al servicio de consulta de datos de usuario por parte de la aplicación.

Figura 49. Clases del controlador del backend (EpsilonBE)

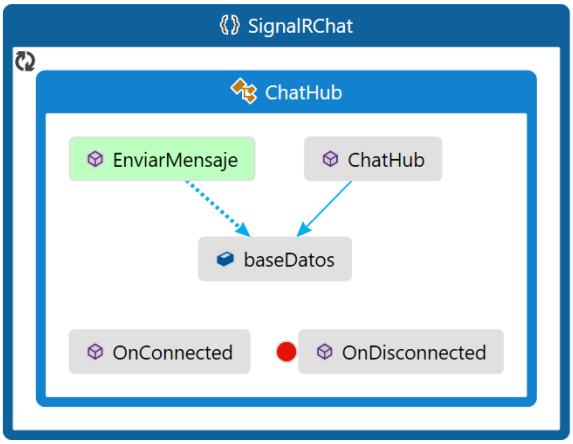


Fuente: Autores del proyecto.

6.5.4 Bus de mensajería (EpsilonHub). Es la plataforma que se encarga de la recepción y envío de mensajes al interior de la aplicación. Su estructura se puede observar en la Figura 50.

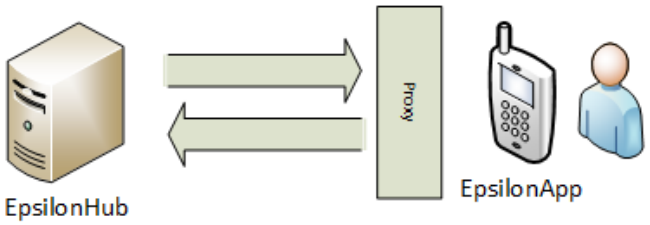
Está implementado sobre la librería SignalR de Microsoft …ver capítulo 4.3…, que brinda funcionalidad de tiempo real para que el servidor pueda enviar mensajes a la aplicación, sin que esta deba estar solicitándolos continuamente. Por lo que la aplicación recibirá información de este componente una vez necesite notificarle de un nuevo mensaje o conocer su estado en tiempo real. El esquema de esta comunicación se puede observar en la Figura 51.

Figura 50. Estructura del componente de mensajería



Fuente: Autores del proyecto.

Figura 51. Comunicación con el bus de mensajería



Fuente: Autores.

Luego de la autenticación del usuario, la aplicación envía una solicitud de conexión a través del proxy (SignalR / *hubs*). En la Figura 52 se presenta un fragmento del código que prepara la conexión al bus y el método para enviar un mensaje al servidor.

Figura 52. Fragmento de código para conexión de la app con el *hub*

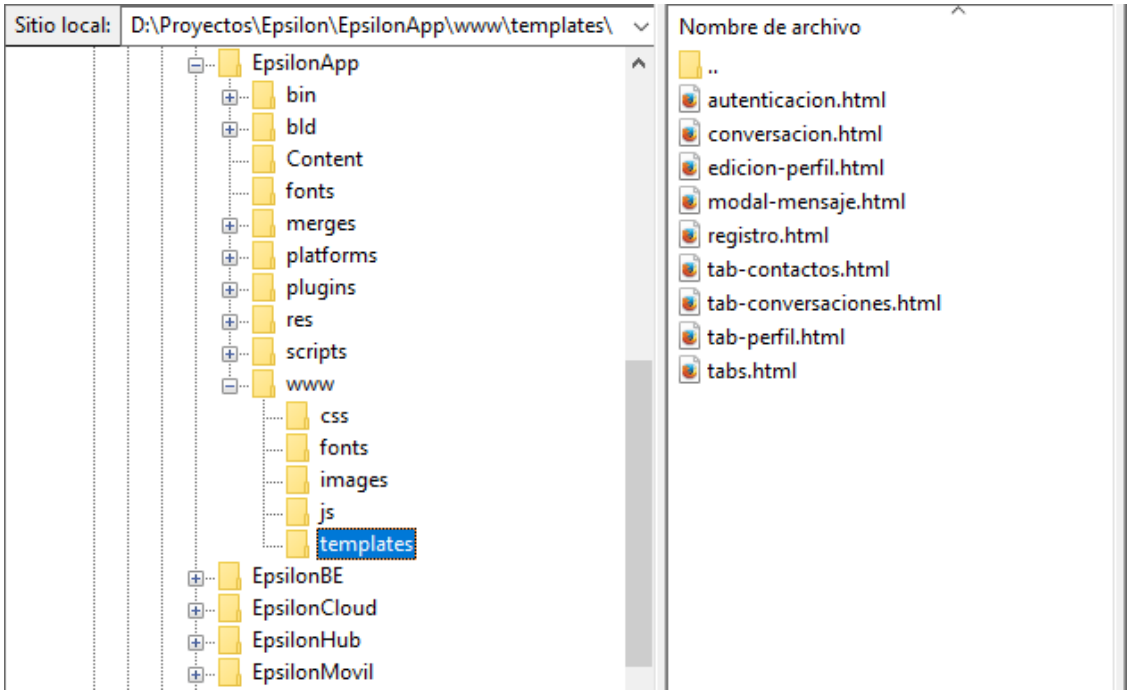
```
//Inicio del bus de mensajería
$.connection.hub.start().done(function () {

    //Método para envío del mensaje
    $scope.enviarMensaje = function () {

        //Servicio de envío de mensaje
        $scope.chatHub.server.enviarMensaje($scope.name, codigoConversacion, $('#texto-mensaje').val());

        //Limpiar variables
        $('#texto-mensaje').val('');
        $scope.message = '';
    };
```

Fuente: Autores del proyecto.

Al recibir la solicitud, el *hub* ejecuta el método de conexión, creando el registro en la tabla de conexiones de la base de datos. Al enviar un mensaje, se llama el método *enviarMensaje* en el proxy. El servidor nuevamente recibe la solicitud, la procesa, enviando al *backend* (*EpsilonBE*) para ocultar el mensaje en una imagen y crear el registro en la base de datos y finalmente, envía la ruta de la imagen al destinatario.

6.5.5 Aplicación móvil (EpsilonApp). Es el componente que se instala en los dispositivos móviles, está construida sobre Apache Cordova, con AngularJS y Ionic, y está compuesta por archivos HTML, JavaScript y estilos CSS. En la Figura 53 se presenta una captura de pantalla con los directorios que la componen.

Figura 53. Estructura del proyecto para la aplicación móvil

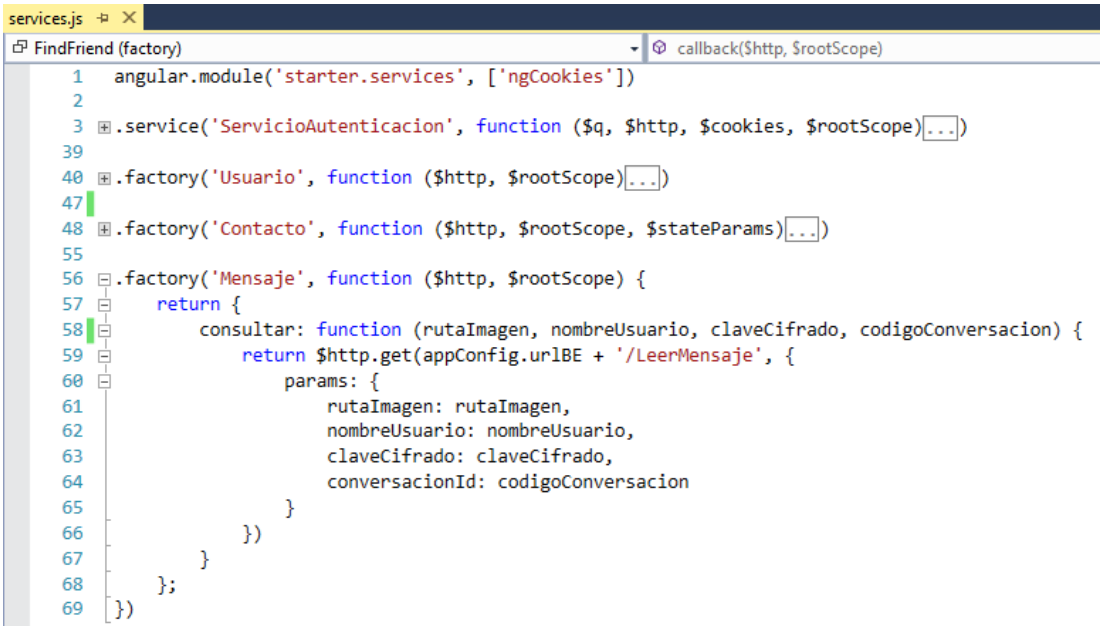


Fuente: Autores del proyecto (captura de explorador)

La lógica se maneja a través de JavaScript, entre estos se tienen los archivos app.js, controller.js y services.js, encargados de la gestión de estados, el control para cada página *HTML* (*template*) y el manejo de las peticiones o servicios al exterior, respectivamente.

Para este último se tienen los componentes presentados en la Figura 54, en la que se destaca el servicio de Mensaje, encargado de enviar la petición al servidor de control EpsilonApp con los parámetros correspondientes.

Figura 54. Estructura del archivo services.js



Fuente: Autores del proyecto (captura de pantalla en Visual Studio)

6.6 ESTRUCTURA DE LA APLICACIÓN

6.6.1 Registro y autenticación. La aplicación presenta a su inicio una pantalla de autenticación, con la opción de registro en caso de ser nuevo y que se presenta en la Figura 55. Para este último caso, solicita los siguientes datos:

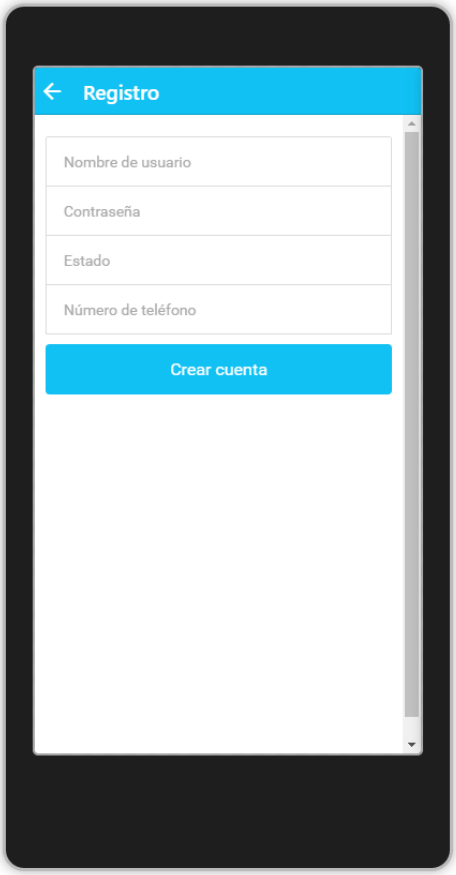
• Nombre de usuario: el sistema valida como único, ya que será el identificador de la persona dentro de la aplicación.

• Contraseña: clave de entrada a la aplicación.

• Estado: una frase para comunicar el estado o situación en el que se presenta (ocupado, en vacaciones, etc.).

• Número de teléfono: para versiones futuras permitiría la validación de número único y la confirmación a través de código por mensaje de texto.

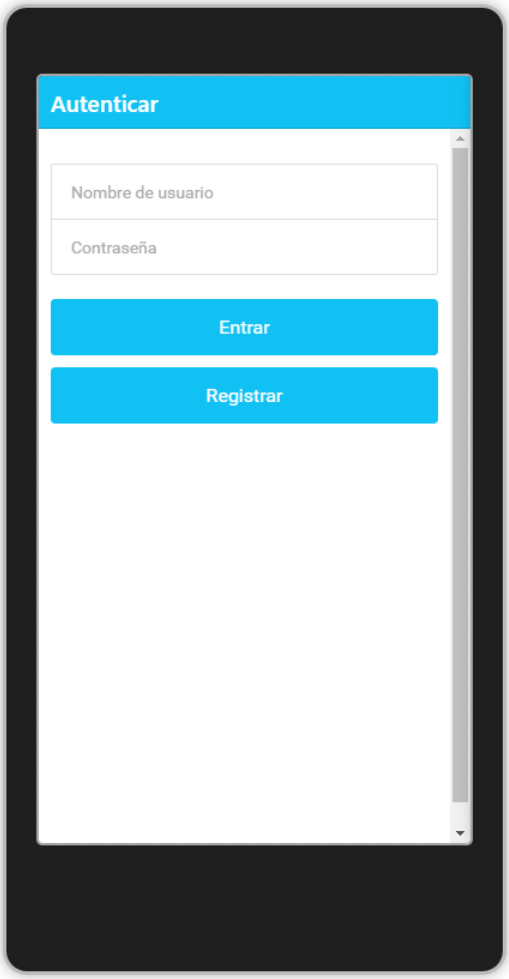
Figura 55. Pantalla de registro



Fuente: Autores del proyecto (captura desde el emulador Ripple).

Para la autenticación, el usuario debe digitar su usuario y contraseña, tal como muestra la Figura 56. El sistema validará los datos contra los almacenados en base de datos (*hash*) y permitirá o no la entrada según su resultado.

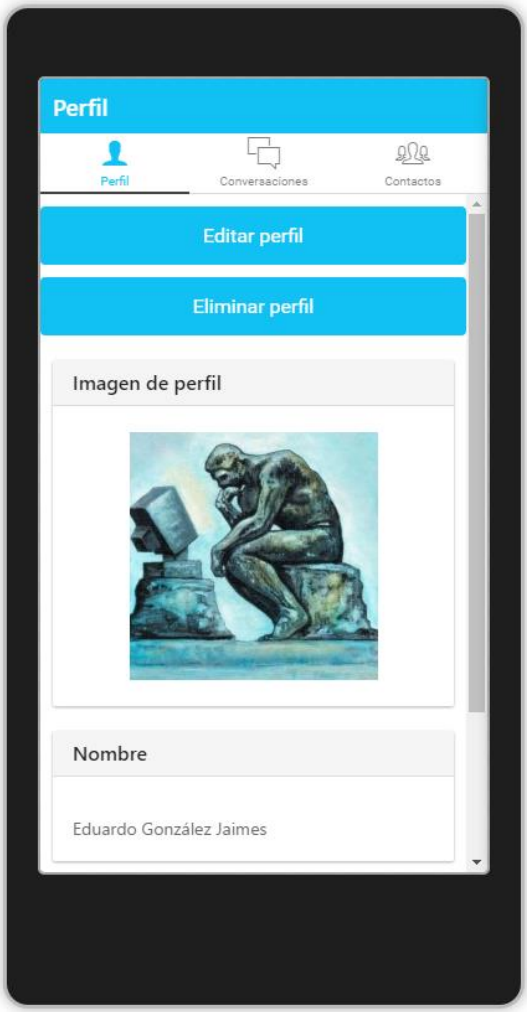
Figura 56. Pantalla de autenticación



Fuente: Autores del proyecto (captura desde el emulador Ripple).

6.6.2 Perfil. Al ingresar a la aplicación, el usuario llegará al componente de perfil, en cual podrá editar o eliminar su cuenta. Además, observará un menú superior en el que tendrá la opción de ingresar a sus conversaciones o a sus contactos. La pantalla de este componente se presenta en la Figura 57.

Figura 57. Perfil del usuario

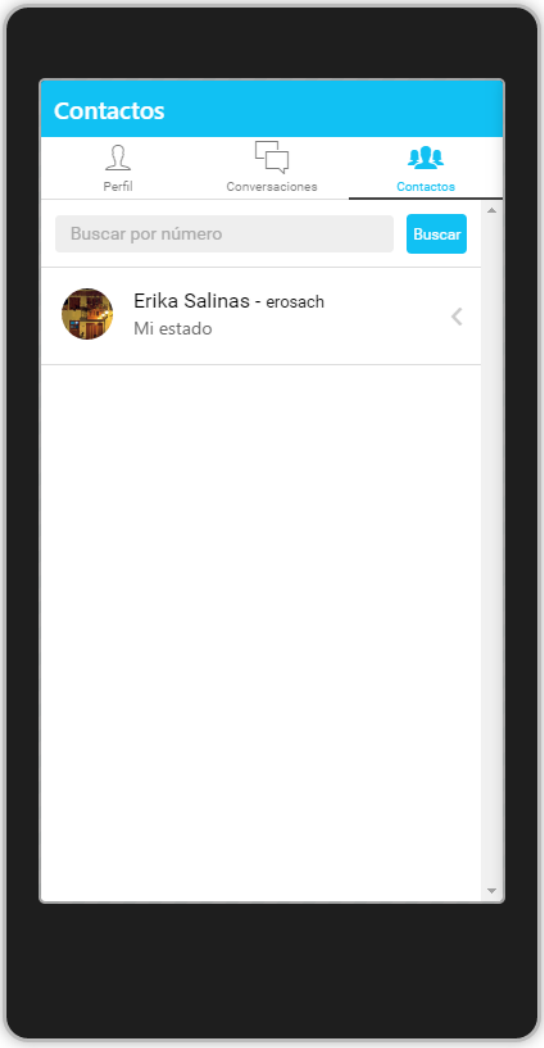


Fuente: Autores del proyecto (captura desde el emulador Ripple).

6.6.3 Contactos. Al seleccionar esta opción, el usuario tendrá la alternativa de ver los contactos con los que podrá conversar o la opción de buscar alguno por su número de teléfono y agregarlo si desea. En la Figura 58 se observa la pantalla implementada para este componente.

Cada uno de los contactos en la lista, tendrá elección para eliminar o iniciar una conversación.

Figura 58. Lista de contactos del usuario



Fuente: Autores del proyecto (captura desde el emulador Ripple).

6.6.4 Conversaciones. En esta pantalla, que se muestra en la Figura 59, se presentan todas las conversaciones que han tenido los contactos entre sí. Al dar clic sobre alguna de las filas con los nombres, el sistema presentará la ventana de conversación con los mensajes previos que se hayan enviado.

Figura 59. Ventana de conversación con mensajes

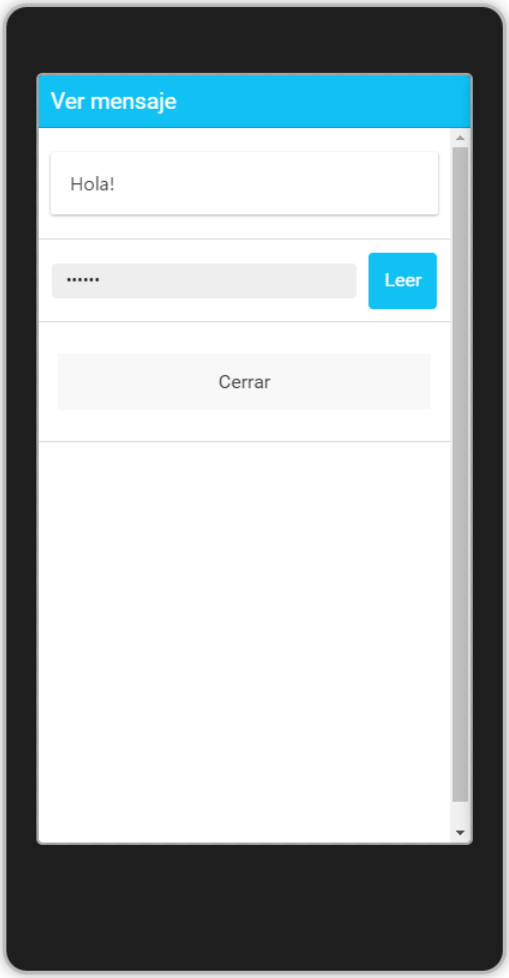
Fuente: Autores del proyecto (captura desde el emulador Ripple).

Para enviar un mensaje, el usuario debe digitar en el cuadro de texto de la parte inferior y, posteriormente, presionar el ícono de la flecha en la parte derecha.

En este momento, la aplicación internamente seleccionará una imagen al azar del repositorio, e incrustará el mensaje escrito en ella. Luego la almacenará y la presentará en la ventana de conversación.

Para visualizar un mensaje, el usuario debe hacer clic sobre la imagen y digitar la clave para descifrar. Esta clave no será visible en el teclado. Una vez digitada y validada, se mostrará el texto en formato plano para su lectura, tal como se muestra en la Figura 60.

Figura 60. Ver mensaje enviado o recibido (oculto)



Fuente: Autores del proyecto (captura desde el emulador Ripple).

6.7 IMPLEMENTACIÓN DE SEGURIDAD EN LOS MENSAJES

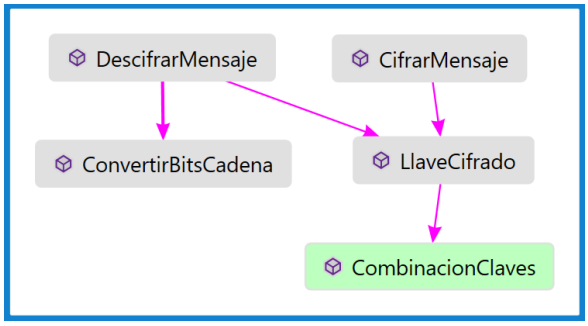
Se contempla la combinación de los dos métodos, esteganografía y cifrado, con el fin de blindar de la mejor y más eficiente forma, los mensajes que se intercambian a través de la aplicación. El mensaje se procesa a través de un método de cifrado (criptografía) y esta nueva cadena se incrusta en la imagen (esteganografía). Para la lectura del mismo, se procede a extraer la cadena cifrada y luego ejecutar un método inverso para descifrarlo y obtener el mensaje plano.

6.7.1 Criptografía.

6.7.1.1 Elección del algoritmo. Se selecciona el método AES, por su uso actual y dificultad en encontrar la cadena plana (una de las formas es mediante fuerza bruta y con un largo consumo de procesamiento y tiempo) y soporte en las librerías propias de *.NET* con su componente *System.Security.Cryptography*[25].

Para su implementación, se crean los métodos *CifrarMensaje* y *DescifrarMensaje* que, junto con sus clases auxiliares, serán los encargados de procesar el mensaje plano en el momento en el que llegue al servidor, así como cuando va a ser entregado nuevamente al dispositivo móvil. Los métodos mencionados, junto con sus auxiliares, se pueden observar en la Figura 61.

Figura 61. Métodos encargados del cifrado del mensaje



Fuente: Autores del proyecto.

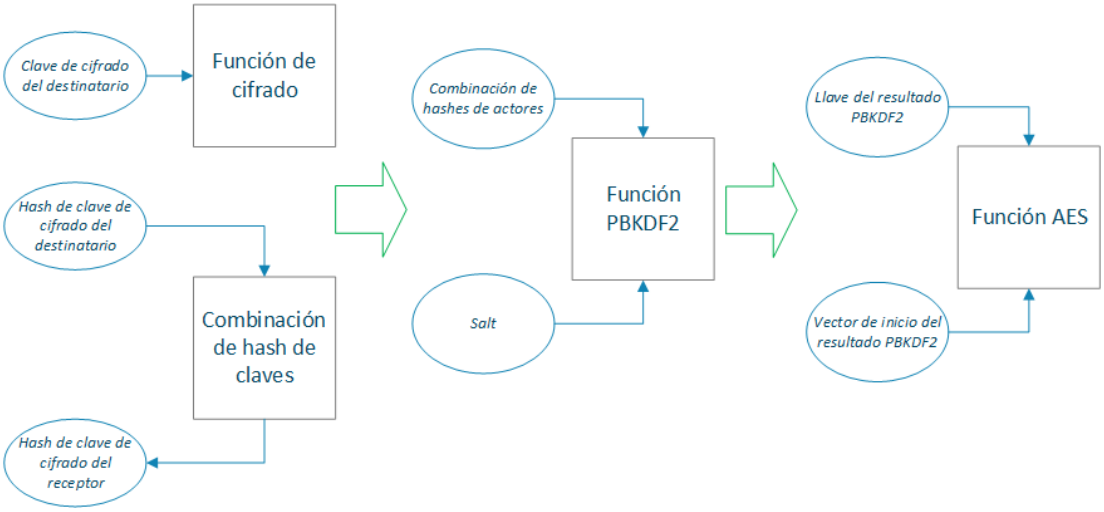
Para el cifrado del mensaje se utiliza una llave formada por una función clave derivada de *PBKDF2*, que utiliza *SHA-1*, y es provista por la librería *Rfc2898DeriveBytes*[26], tomando como valores de entrada la contraseña de cifrado y un valor semilla (*salt*) predefinido.

Previamente el *hash* de la clave de cifrado del destinatario se ha combinado en un orden determinado con el *hash* de la clave del receptor. Asegurando que tanto el uno como el otro, puedan leer los mensajes que intercambian con el uso de sus

[25] MICROSOFT NETWORK. Aes Class. {En línea}. {Febrero de 2017} Disponible en https://msdn.microsoft.com/en-us/library/system.security.cryptography.aes(v=vs.110).aspx
[26] MICROSOFT NETWORK. Rfc2898DeriveBytes Class. {En línea}. {Febrero de 2017} Disponible https://msdn.microsoft.com/en-us/library/system.security.cryptography.rfc2898derivebytes.aspx

respectivas claves. El diagrama de flujo de este procedimiento se observa en la Figura 62.

Figura 62. Proceso de cifrado del mensaje



Fuente: Autores del proyecto.

6.7.2 Esteganografía.

6.7.2.1 Elección de técnica. Para la implementación del módulo de esteganografía en la aplicación, se eligió la técnica de sustitución de bits descrita anteriormente… Ver capítulo 4.10.1.1…

Para llegar a esta decisión, se analizó un factor crítico como el tamaño del archivo contenedor del mensaje, debido al intercambio constante de información entre el servidor y la aplicación instalada en el dispositivo móvil. Al sustituir los bits de la imagen seleccionada para contener el texto, por los propios del mensaje, se asegura que el tamaño se mantiene y la calidad de la imagen no difiere en gran medida de la original.

6.7.2.2 Descripción del proceso. Teniendo como referencia el proyecto *Steganography*: *Simple Implementation in* C#[27], se ejecutan las siguientes fases para ocultar y extraer el mensaje.

[27] SOBOH, H. Steganography: Simple Implementation in C#. {En línea}. {Febrero de 2017}. Disponible en: https://www.codeproject.com/Tips/635715/Steganography-Simple-Implementation-in-Csharp

Una vez la aplicación, en el lado del servidor, recibe el mensaje y consulta al repositorio para recoger la imagen original en la que lo va a ocultar, realiza los siguientes pasos:

- Recorrer los pixeles de la imagen y convertir a 0 el bit menos significativo de cada valor binario RGB. Para el ejemplo, se observa el pixel RGB (177,182, 201), encontrando un valor en la propiedad rojo (R) del pixel con valor de 177. Este se convertirá a binario a 10110001, el valor verde (G) equivalente a 10110110 seguirá siendo el mismo, ya que su bit menos significativo es 0. Por último, el valor azul (B) de 201, 11001001 en binario, será transformado a 11001001, quedando finalmente un nuevo pixel con el valor RGB (176,182,200).

- Convertir el carácter correspondiente de la cadena de texto a ocultar, a su valor entero. En el ejemplo realizado, se observa que la primera letra del mensaje “Prueba” corresponde a la P, con un valor ASCII de 80, 01010000 en binario.

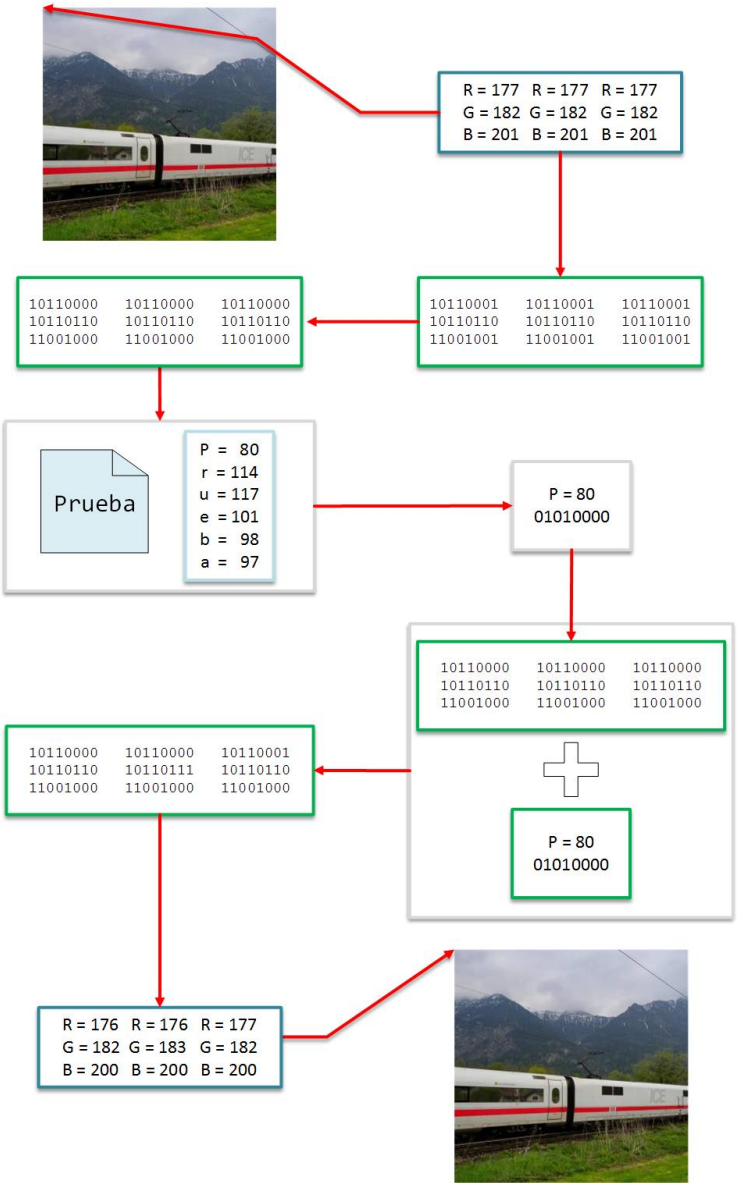
- Ocultar los 8 bits del carácter en los 8 primeros bits de los pixeles (3 en los correspondientes al rojo, 3 en los del verde y 2 los del azul). En este apartado se utilizan 3 pixeles, lo que corresponde a 9 elementos para ocultar cada carácter. En ASCII los valores ocupan 8 posiciones, por lo que el primer carácter se ocultará en los valores RGB de los dos primeros pixeles y en los valores R y G del tercer pixel.

- Repetir el procedimiento con el siguiente carácter en el próximo pixel, ocupando la posición que quedó libre de los caracteres anteriores.

- Al finalizar de procesar todos los caracteres de la cadena, se oculta un indicador, para que el proceso de extracción conozca hasta dónde debe ir en el análisis de cada pixel.

En la Figura 63 se presentan los pasos descritos, mostrando cómo ocultaría el primer carácter (P) del mensaje “Prueba” dentro de una imagen seleccionada aleatoriamente.

Figura 63. Pasos a seguir para ocultar el mensaje en una imagen



Fuente: Autores del proyecto.

A continuación, se describe el procedimiento para extraer el mensaje de una imagen que lo contiene:

- Recorrer los pixeles de la imagen cifrada.

- Extraer los bits correspondientes a los caracteres ocultos, teniendo en cuenta los cambios en los bits menos significativos (0 ó 1, según corresponda).

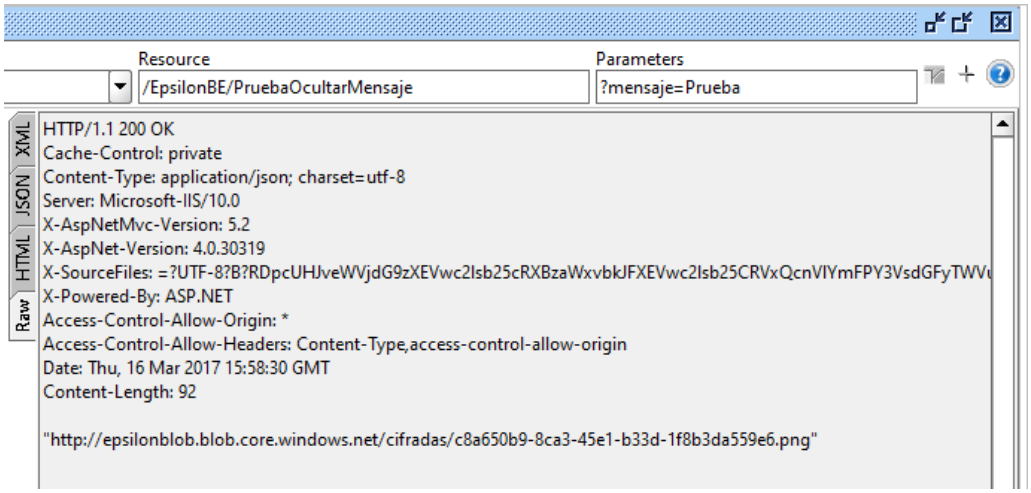
• Almacenar los dígitos binarios encontrados hasta llegar a la longitud de 8, convirtiendo luego esta cifra en su equivalente ASCII para conocer el carácter correspondiente al mensaje.

• Detener la tarea cuando se encuentre el indicador de fin de texto y entregar el mensaje oculto.

6.7.3 Ejecución. Se va a ejecutar el proceso de esteganografía invocando un método de prueba creado para tal fin. Para esto se envía el mismo mensaje "Prueba" al método señalado:

Al enviar una petición al servicio, se recibe como respuesta la ruta de la imagen cifrada. En la Figura 64, se puede ver una captura de pantalla con la respuesta de una petición enviada para ocultar un mensaje.

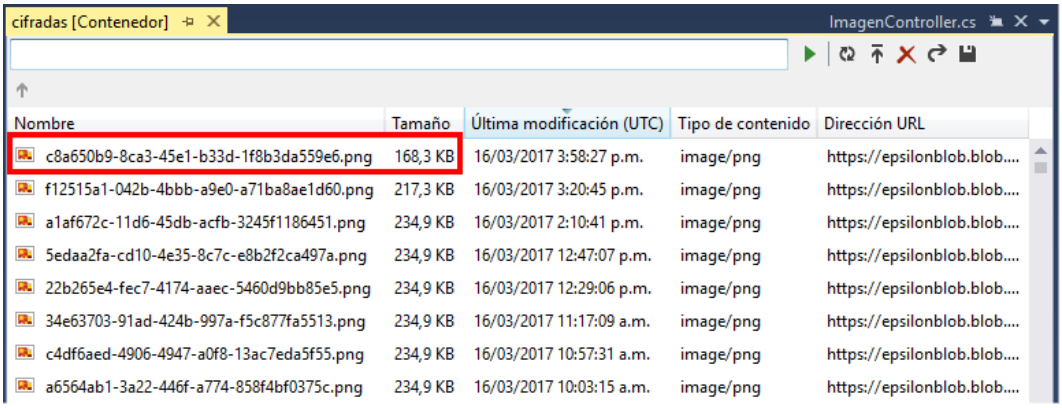
Figura 64. Ejecución del servicio de prueba para ocultar el mensaje



Fuente: Autores del proyecto (captura de pantalla de SoapUI).

El sistema aleatoriamente selecciona una imagen del repositorio y oculta en ella el mensaje enviado como parámetro, creando una nueva que posteriormente carga en el directorio de imágenes cifradas del mismo repositorio. En la Figura 65 se muestra la ruta de la imagen que contiene el mensaje previamente procesado por el servicio de ocultar.

Figura 65. Imagen cifrada en repositorio e imagen original



Fuente: Autores del proyecto (captura de pantalla desde Visual Studio).

Si se verifica al interior del código, se puede observar un algoritmo similar al de la Figura 66, que indica el paso a paso por el proceso de ocultar el mensaje: buscar una imagen contenedora aleatoria, ocultar el mensaje, guardar la nueva imagen en el repositorio y responder con la ruta de almacenamiento.

Figura 66. Pasos para ocultar el mensaje en el método *ProcesarImagen*

```
2 referencias
public static string ProcesarImagen(string mensaje)
{
    //Asignación de variable para la ruta de la imagen seleccionada de forma aleatoria
    string rutaImagen = ImagenAleatoria();

    //Lectura de la imagen original, según la ruta elegida
    Image imagenOriginal = ImagenController.LeerImagen(rutaImagen);

    //Oculta el mensaje, obteniendo una nueva imagen
    Image imagenConvertida = EsteganografiaController.GuardarMensaje(mensaje, imagenOriginal);

    //Carga de la nueva imagen en el repositorio
    string rutaNuevaImagen = AlmacenamientoController.subirImagen(imagenConvertida);

    //Respuesta de la ruta de la nueva imagen
    return rutaNuevaImagen;
}
```

Fuente: Autores del proyecto (captura de pantalla en Visual Studio).

El proceso de almacenar la imagen se hace luego de convertir el formato de la misma de PNG a BMP, para poder hacer la comparación en detalle de los dos archivos luego de esa ejecución, sin conversiones ni trabajos adicionales. En la Figura 67 se observan los fragmentos de código que se encargan de este proceso. Se crearon dos líneas adicionales temporalmente para almacenar los archivos en el disco.

Figura 67. Fragmentos del código para almacenar las imágenes temporales

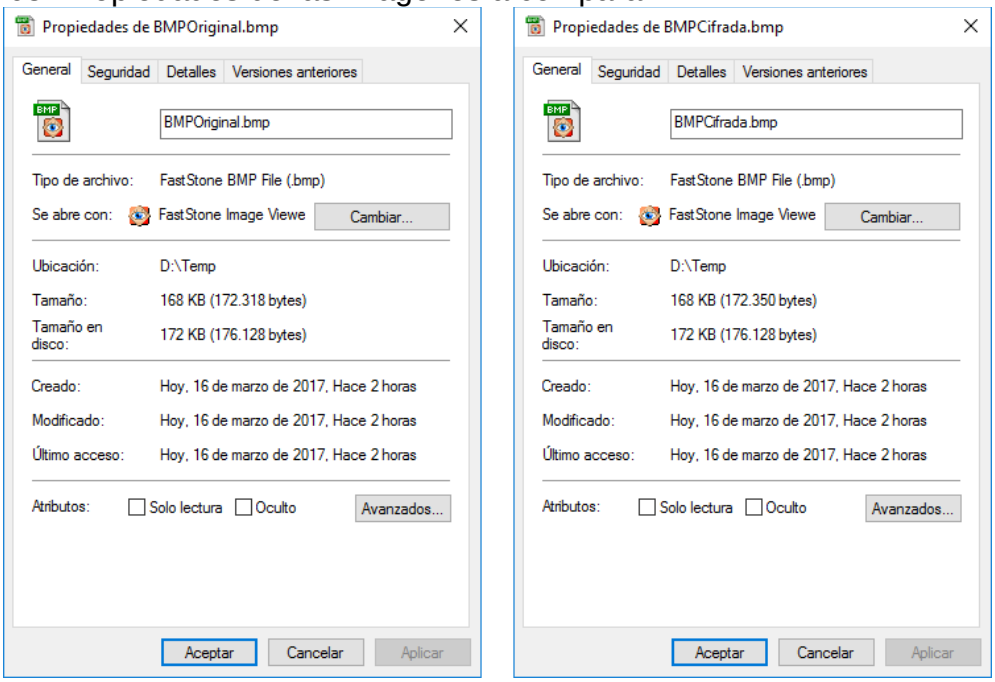
```
//Creación del bmp a partir de la imagen y asignación del estado de ocultar
Bitmap bmp = new Bitmap(imagen);

bmp.Save("D:\\Temp\\BMPOriginal.bmp");

Estado estado = Estado.ocultar;

//Asignación de posiciones iniciales
int charIndex = 0;
int charValue = 0;
long colorUnitIndex = 0;
int zeros = 0;
int R = 0, G = 0, B = 0;
```

```
//Recorrido por cada conjunto de bits según el color (RGB)
for (int n = 0; n < 3; n++)
{
    //Validación de finalización de bits contenedores para el pixel (8)
    if (colorUnitIndex % 8 == 0)
    {
        //Validación de indicador de finalización de cadena a ocultar
        if (zeros == 8)
        {
            if ((colorUnitIndex - 1) % 3 < 2)
            {
                bmp.SetPixel(j, i, Color.FromArgb(R, G, B));
            }

            //Responder con la nueva imagen
            bmp.Save("D:\\Temp\\BMPCifrada.bmp");
            return bmp;
        }
```

Fuente: Autores del proyecto (captura de pantalla desde Visual Studio).

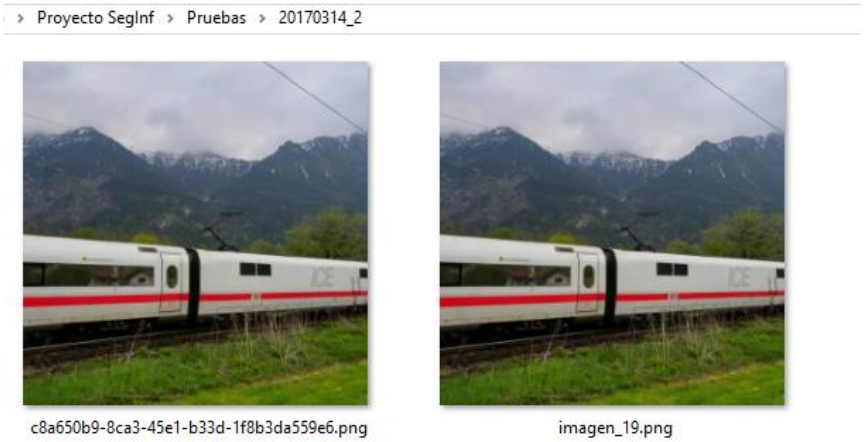
De esta manera, se obtienen las dos imágenes con propiedades similares y percepción visual idéntica, tal como se presenta en la Figura 68. En la Figura 69, se muestra que, al comparar las imágenes, no se observan diferencias a simple vista.

Figura 68. Propiedades de las imágenes a comparar



Fuente: Autores del proyecto (captura desde Windows 10).

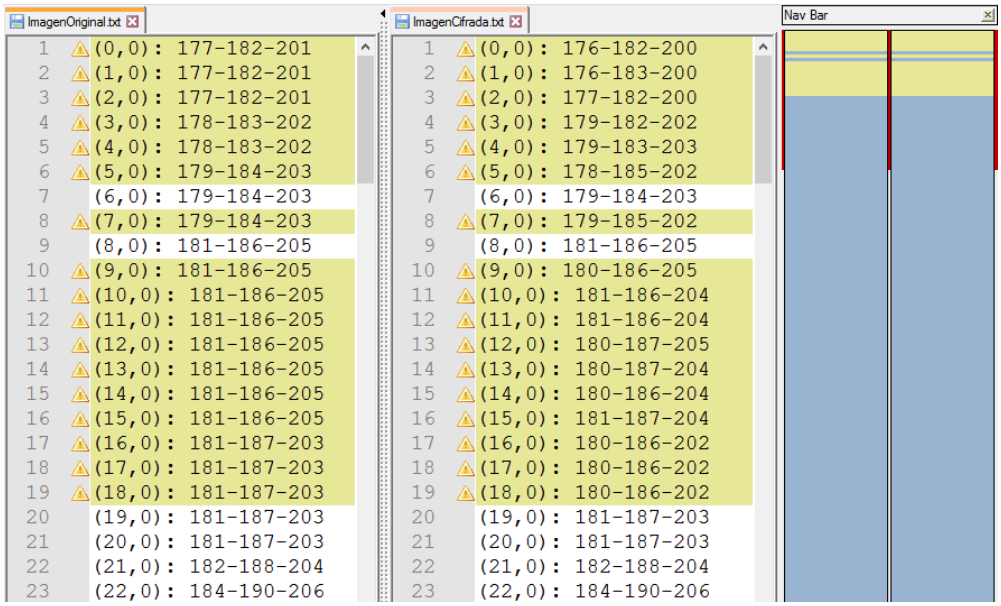
Figura 69. Comparación de la imagen original con la que oculta el mensaje



Fuente: Autores del proyecto (captura de pantalla en Windows 10).

Al verificar los bits de cada imagen, se observa una diferencia entre una y otra en los primeros pixeles (que son los que contienen el mensaje). De esta manera, se encuentra que se modificaron las primeras 19 líneas. En la Figura 70, se tienen dos líneas que no se modificaron (líneas 7 y 8), esto se presenta, porque al cambiar un bit menos significativo de 1 a 0, se puede obtener de la cadena de texto a guardar nuevamente un bit correspondiente a 1, por lo que este pixel va a quedar igual al original.

Figura 70. Comparación de valores RGB de las dos imágenes



Fuente: Autores del proyecto (captura de pantalla de Notepad++).

6.8 EVALUACIÓN DE SEGURIDAD DE LA APLICACIÓN

6.8.1 Análisis de cumplimento de requerimientos. Teniendo como base los requerimientos de seguridad planteados en la fase de análisis del proyecto, se ejecutó un plan de pruebas que permitiera el asegurar el cumplimiento de los mismos:

6.8.1.1 Lectura de mensajes. Los mensajes que se encuentran ocultos en una imagen, podrán ser visualizados por el usuario si digita una clave. De esta manera se asegura la confidencialidad de la información en caso que un intruso pueda tener acceso al dispositivo móvil.

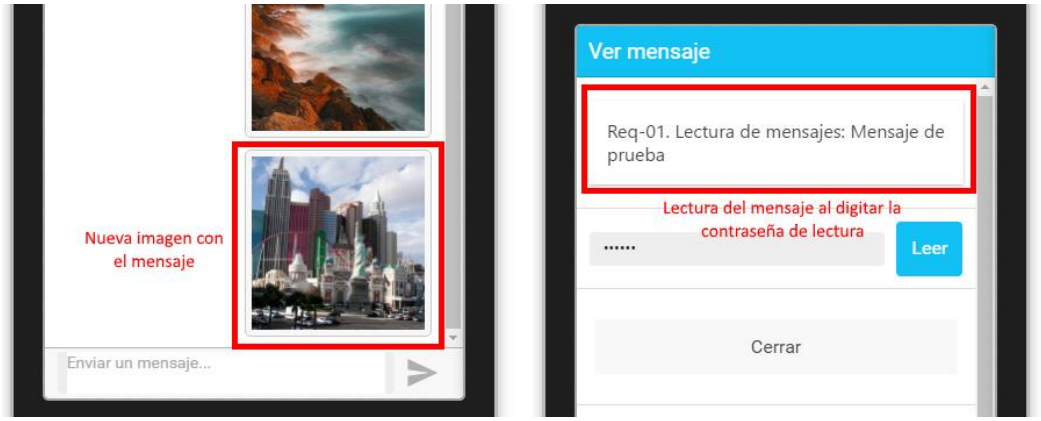
En la Figura 71, se observa cómo se envía un mensaje de prueba y, posteriormente, se revisa si en el dispositivo del destinatario se observa una nueva imagen correspondiente al mensaje enviado.

Figura 71. Envío del mensaje de prueba

Fuente: Autores del proyecto (captura de pantalla del emulador Ripple).

En la Figura 72 se ve que la imagen llega al dispositivo del destinatario, por lo que se debe proceder a la lectura del mensaje, haciendo clic sobre ella y digitando la contraseña de lectura.

Figura 72. Nueva imagen y lectura del mensaje

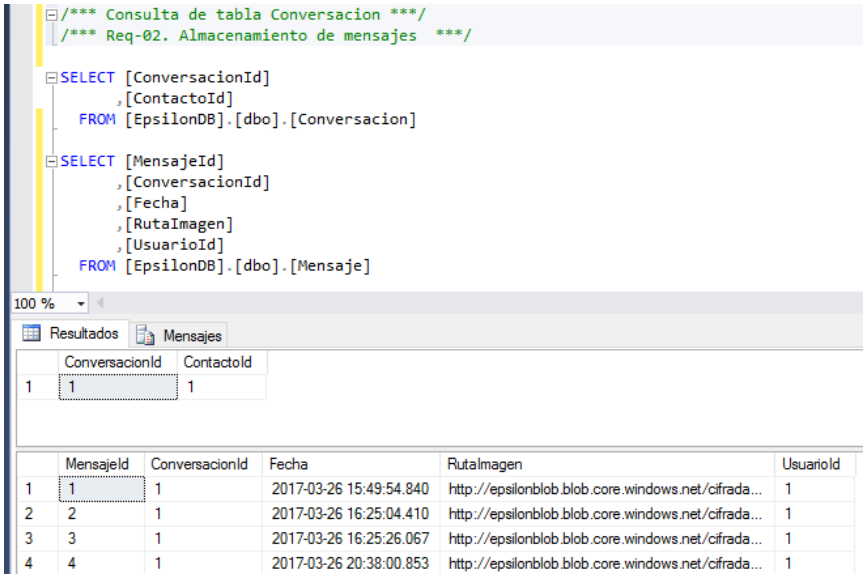


Fuente: Autores del proyecto (captura de pantalla del emulador Ripple).

6.8.1.2 Almacenamiento de mensajes. Los mensajes no deben almacenarse en la base de datos. De tal forma que, si un intruso accede a esta, no pueda observar los textos de cualquier conversación.

Se examina la base de datos. Revisando los campos en las tablas *Conversacion* y *Mensaje*, encargadas de almacenar los registros de una conversación entre dos contactos, tal como se muestra en la Figura 73.

Figura 73. Consulta en base de datos (*Conversacion* y *Mensaje*)



```
/*** Consulta de tabla Conversacion ***/
/*** Req-02. Almacenamiento de mensajes  ***/

SELECT [ConversacionId]
      ,[ContactoId]
  FROM [EpsilonDB].[dbo].[Conversacion]

SELECT [MensajeId]
      ,[ConversacionId]
      ,[Fecha]
      ,[RutaImagen]
      ,[UsuarioId]
  FROM [EpsilonDB].[dbo].[Mensaje]
```

100 %

Resultados | Mensajes

	ConversacionId	ContactoId
1	1	1

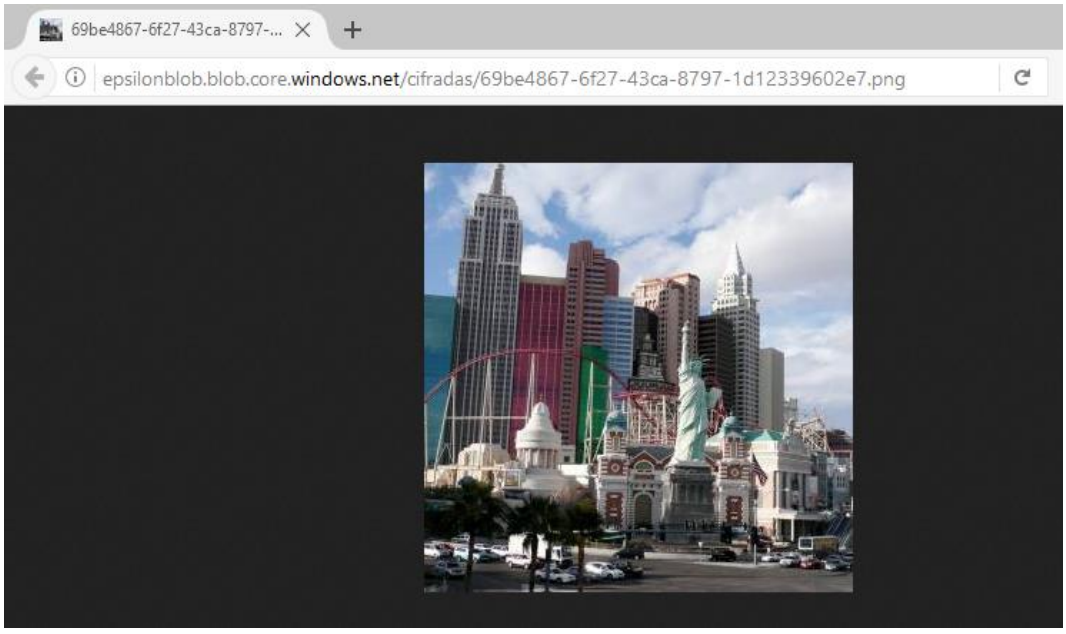
	MensajeId	ConversacionId	Fecha	RutaImagen	UsuarioId
1	1	1	2017-03-26 15:49:54.840	http://epsilonblob.blob.core.windows.net/cifrada...	1
2	2	1	2017-03-26 16:25:04.410	http://epsilonblob.blob.core.windows.net/cifrada...	1
3	3	1	2017-03-26 16:25:26.067	http://epsilonblob.blob.core.windows.net/cifrada...	1
4	4	1	2017-03-26 20:38:00.853	http://epsilonblob.blob.core.windows.net/cifrada...	1

Fuente: Autores del proyecto (desde SQL Server Management Studio).

En la tabla *Mensaje*, se encuentra la ruta de la imagen. Lo cual muestra que no se está guardando el mensaje como tal, sino la ubicación en el repositorio de la imagen que lo contiene. Si algún tercero logra acceder a esta base de datos, no tendrá disponibilidad de visualizar los textos de la conversación y deberá posteriormente encontrar el mensaje oculto y descifrarlo.

6.8.1.3 Análisis sobre la técnica de esteganografía. Al consultar una de las rutas de la consulta anterior, se puede descargar la imagen, tal como se puede ver en la Figura 74. A este archivo se le hace un análisis con tres herramientas de esteganografía libres en la red para tal fin.

Figura 74. Descarga de imagen del repositorio

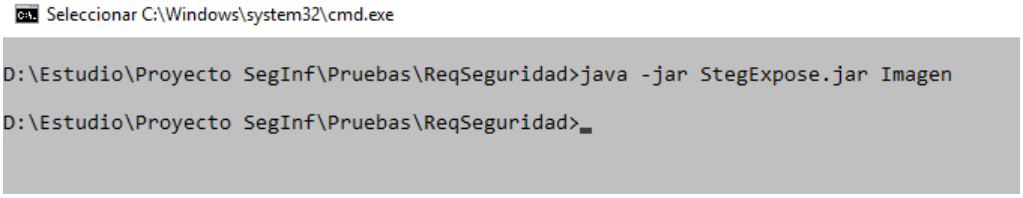


Fuente: Autores del proyecto.

Al analizar con *StegExpose*[28], no se encuentra ningún mensaje sospechoso en la imagen. En la Figura 75 se puede observar una captura de la consola con la ejecución de la herramienta, sin ninguna respuesta de posible mensaje oculto.

[28] BOEHM, B. StegExpose. {En línea}. {Enero de 2017}. Disponible en: https://github.com/b3dk7/StegExpose

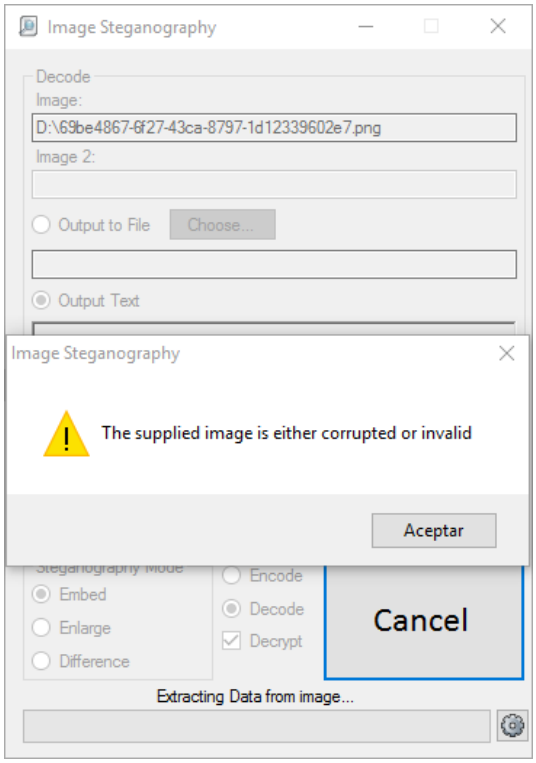
Figura 75. Análisis con *StegExpose* de la imagen



Fuente: Autores del proyecto (captura de pantalla de consola en Windows).

Si se analiza la imagen con otra herramienta, *Image Steganography*[29], tampoco se obtiene un resultado satisfactorio de hallazgo de información oculta o indicio de anormalidad. Es importante destacar que el hecho de que la imagen esté en formato PNG, lo cual dificulta la tarea de las herramientas más comunes, tal como se observa en la Figura 76.

Figura 76. Análisis con *Image Steganography* de la imagen

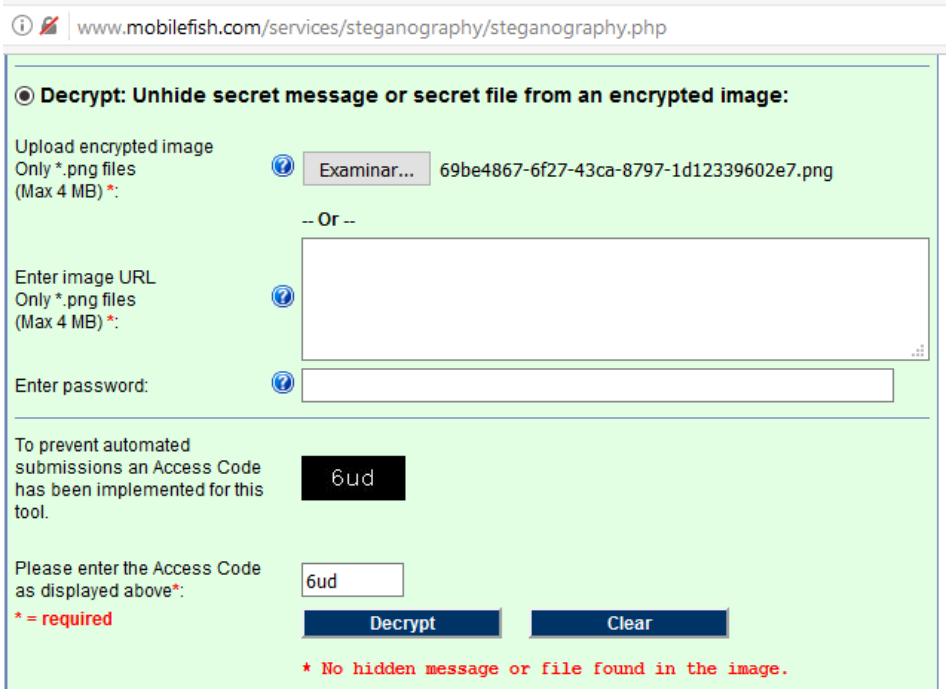


Fuente: Autores del proyecto (captura de pantalla de la herramienta).

[29] STANLEY, J. Image Steganography. {En línea}. {Enero de 2017}. Disponible en http://incoherency.co.uk/image-steganography/

Por último, se revisa la imagen con una herramienta de análisis en línea (*MobileFish Online steganography service*[30]). En la Figura 77 se observa que tampoco se obtiene evidencia de un mensaje oculto.

Figura 77. Análisis de la imagen en *MobileFish*

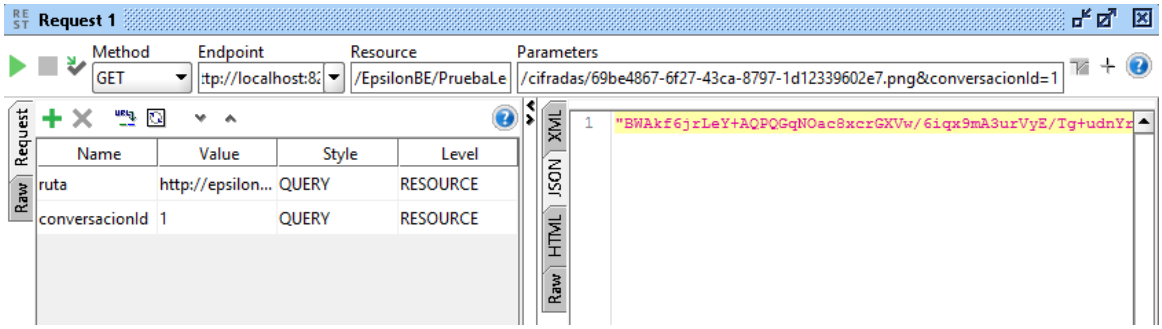


Fuente: Autores del proyecto (captura de pantalla).

6.8.1.4 Análisis sobre la técnica de criptografía. Debido a que ninguna de las herramientas de estego-análisis utilizadas encontró el mensaje, se utiliza el servicio web de prueba para extraer el mensaje oculto, enviando una petición directa al método de lectura del mensaje con la ruta, el código de la conversación y la clave de lectura. La Figura 78 muestra una captura de pantalla con una petición al servicio de prueba que extrae el texto cifrado de la imagen con el mensaje oculto.

[30] MOBILE FISH. Online steganography service. {En línea}. {Enero de 2017}. Disponible en http://www.mobilefish.com/services/steganography/steganography.php

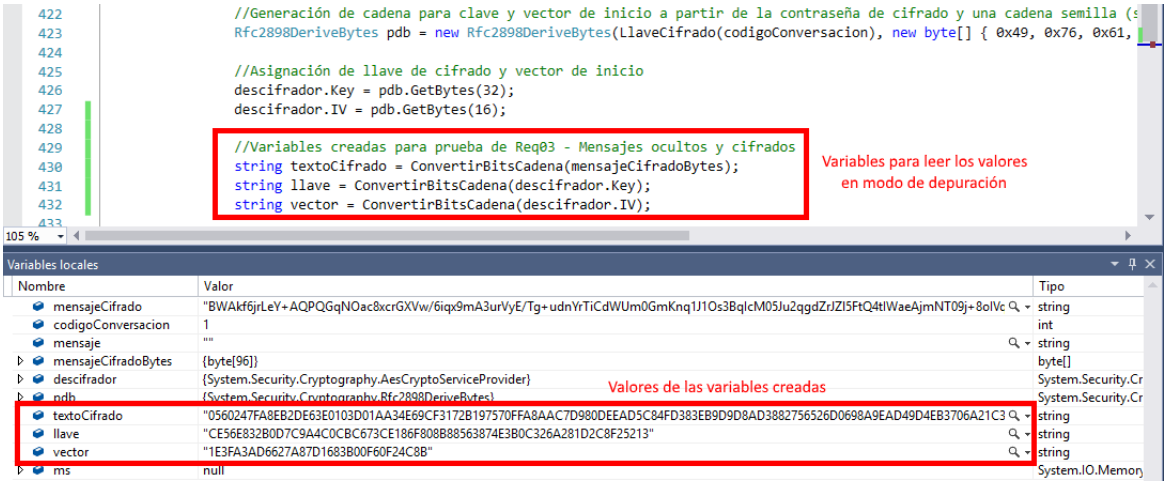
Figura 78. Servicio web para extraer el mensaje oculto en la imagen



Fuente: Autores del proyecto (captura de pantalla de SoapUI).

Para el texto, se tiene un cifrado *AES-256* que asegura, hasta el día de hoy, la confidencialidad del mensaje. Se realiza una prueba para descifrarlo, obteniendo en modo de depuración la llave, el vector de inicio y el mensaje en formato hexadecimal, tal como se observa en la Figura 79.

Figura 79. Creación de variables en código para valores de descifrado



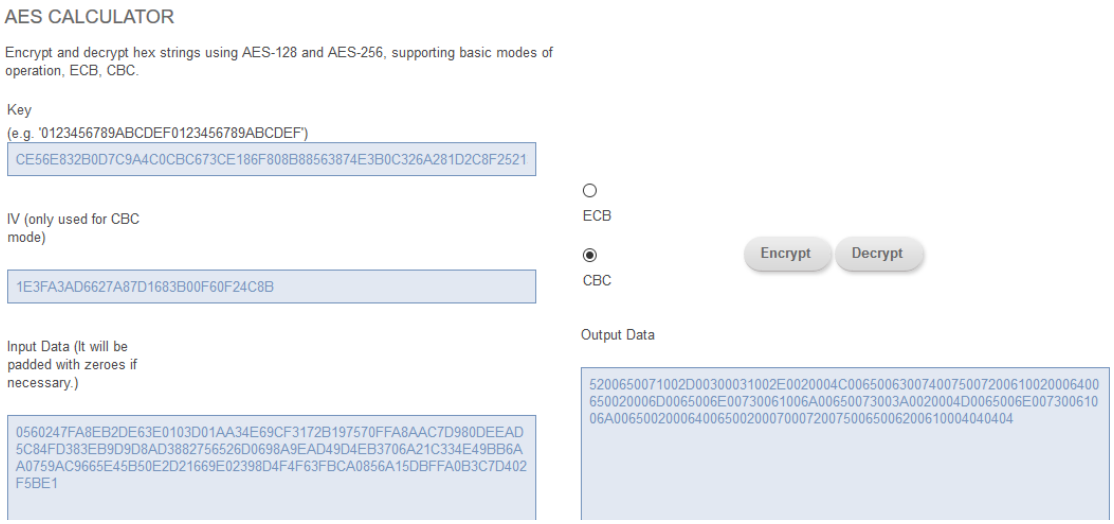
Fuente: Autores del proyecto (captura de pantalla de Visual Studio).

Con estos valores se utiliza una herramienta en línea de descifrado para verificar que el mensaje corresponde al original. En la Figura 80 se observa la ejecución sobre la herramienta *AES Calculator*[31], ingresando los valores hexadecimales

[31] CRYPTOMATHIC. AES Calculator. {En línea}. {Enero de 2017}. Disponible en http://extranet.cryptomathic.com/aescalc

extraídos en el paso anterior y seleccionando la opción CBC para el uso del vector de inicio.

Figura 80. Uso de herramienta en línea para descifrar *AES-256*



Fuente: Autores del proyecto (captura de pantalla del navegador).

Al tomar el texto generado, eliminar los ceros de relleno y convertirlo a ASCII, se puede observar el mensaje original, lo que confirma el éxito de la ejecución del método de cifrado. En la Figura 81, se presenta una captura de la herramienta en línea *Hexadecimal to String*[32] con el mensaje en formato plano.

Figura 81. Conversión de hexadecimal a ASCII

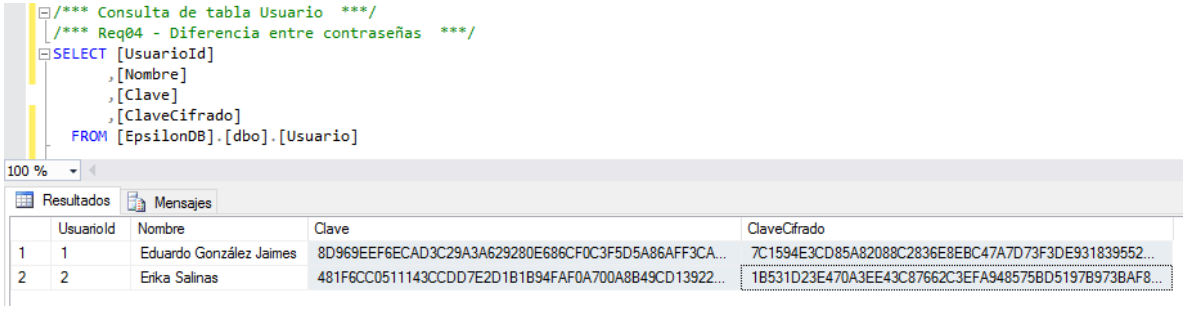
Fuente: Autores del proyecto (captura de pantalla del navegador).

[32] CODE BEAUTIFY. HexaDecimal to String. {En línea}. {Noviembre de 2016}. Disponible en http://codebeautify.org/hex-string-converter

6.8.1.5 Diferencia entre contraseñas. La clave para descifrar mensajes debe ser diferente a la clave de autenticación, para evitar que al conocer o descubrir una de ellas, un tercero pueda acceder directamente a los mensajes.

En la Figura 82 se observa que, al consultar la tabla de *Usuario*, se tienen dos campos para cada *hash* correspondientes a las dos claves requeridas:

Figura 82. Consulta a la tabla Usuario para validación de campos de clave



```
/*** Consulta de tabla Usuario  ***/
 /*** Req04 - Diferencia entre contraseñas  ***/
SELECT [UsuarioId]
      ,[Nombre]
      ,[Clave]
      ,[ClaveCifrado]
  FROM [EpsilonDB].[dbo].[Usuario]
```

100 %

Resultados | Mensajes

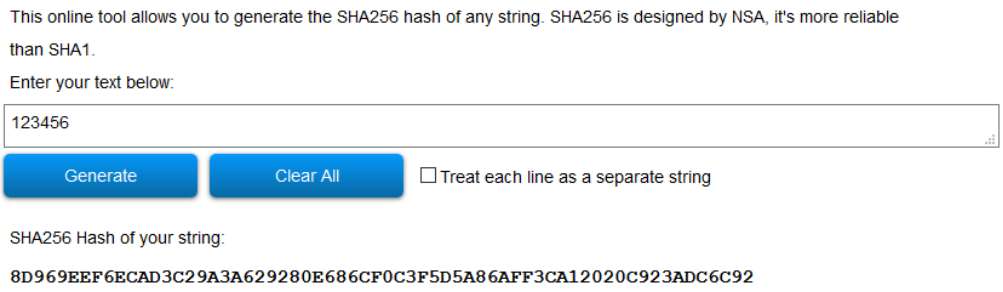
	UsuarioId	Nombre	Clave	ClaveCifrado
1	1	Eduardo González Jaimes	8D969EEF6ECAD3C29A3A629280E686CF0C3F5D5A86AFF3CA...	7C1594E3CD85A82088C2836E8EBC47A7D73F3DE931839552...
2	2	Erika Salinas	481F6CC0511143CCDD7E2D1B1B94FAF0A700A8B49CD13922...	1B531D23E470A3EE43C87662C3EFA948575BD5197B973BAF8...

Fuente: Autores del proyecto (captura de SQL Server Management Studio).

6.8.1.6 Almacenamiento de contraseñas. Tanto la clave de autenticación como la de lectura del mensaje, no deben ser almacenadas en la base de datos de forma plana. Se debe contar con un sistema que no permita a intrusos el conocerla fácilmente, en caso de que logren acceder a la capa de datos.

Para validar este requerimiento, se utiliza una herramienta en línea que obtiene el *hash* de cadenas de texto *SHA256 Hash Generator*, tal como se observa en la Figura 83. De esta forma, se comprobó si una contraseña conocida es almacenada luego de ser convertida a su función de resumen (*hash*).

Figura 83. Función *hash* en línea para verificación de clave



This online tool allows you to generate the SHA256 hash of any string. SHA256 is designed by NSA, it's more reliable than SHA1.

Enter your text below:

123456

Generate | Clear All | ☐ Treat each line as a separate string

SHA256 Hash of your string:

8D969EEF6ECAD3C29A3A629280E686CF0C3F5D5A86AFF3CA12020C923ADC6C92

Fuente: Autores del proyecto (captura de pantalla del navegador).

Al comparar esta cadena con la consultada en base de datos para el primer usuario …ver Figura 57…, se observa que son idénticas, lo que confirma el almacenamiento requerido de la misma.

7 CONCLUSIONES Y RECOMENDACIONES

7.1 CONCLUSIONES

Durante el análisis del aplicativo, se investigaron diferentes métodos de seguridad a implementar, sus características, ventajas y desventajas, y sus técnicas de implementación. De esta forma, se decidió por un diseño que combina esteganografía de imágenes a través del método de sustitución del bit menos significativo, junto con cifrado mediante el algoritmo AES y funciones HASH. Logrando un diseño y posterior implementación que cumple con los requerimientos definidos desde el inicio.

Se exploraron otros elementos de seguridad de la información y varios apartados que implican el desarrollo o implementación de un proyecto de este tipo, junto con los requerimientos de seguridad necesarios: intercambio de peticiones HTTP, almacenamiento en base de datos, validaciones de contraseñas, generación de nombres aleatorios para almacenamiento, peticiones *pull* y *push* entre cliente y servidor, entre otras.

La implementación de esta aplicación, permite concluir que aparte de las consideraciones técnicas y de diseño que se deben tener para procurar un correcto funcionamiento y desempeño de la misma, se deben tener en cuenta elementos de seguridad desde el inicio. Partiendo por una especificación de requerimientos que guíen hacia su cumplimiento una vez se finalice la fase de desarrollo, realizando una revisión de que cada uno de ellos se cumple tal y como se espera.

Teniendo en consideración todo lo mencionado con anterioridad para la realización de una aplicación de este tipo, es factible concluir que las decisiones y actividades efectuadas encaminadas a desarrollarlo y que permitan el alcance planeado (dos usuarios intercambiando información sin descuidar su confidencialidad e integridad), tales como la utilización de un método de sustitución de bits para reemplazar los bits menos significativos en una escala de color determinada o la metodología seleccionada para su implementación logran los objetivos planeados. Esto significa que no sólo se puede enviar información de una manera confiable, sino que también se agrega un nivel de seguridad mayor al que brindan las aplicaciones de mensajería instantánea actualmente.

Aunque el desarrollo de la aplicación cumple con lo determinado en el alcance, existen cosas que se pueden mejorar, como lo es la utilización de otros contenedores para ocultar la información, el personalizar los requerimientos de los

usuarios y el agregar otros métodos esteganográficos o permitir las conversaciones grupales entre contactos que lo requieran.

7.2 RECOMENDACIONES

Los resultados del desarrollo de la aplicación fueron satisfactorios y los mismos se encuentran alineados con el alcance planteado en el proyecto, sin embargo, existe una serie de actividades a llevar a cabo en una posterior versión:

- Implementar los requerimientos de seguridad adicionales descritos en el capítulo correspondiente…Ver numeral 6.2.2...

- Publicar el aplicativo en las tiendas oficiales. De esta manera se podrá instalar y ejecutar de una manera mucho más confiable, rápida y al alcance de todas las personas que lo necesiten.

- Para tener una mejor interacción entre los usuarios, se propone agregar la opción de crear conversaciones grupales o multiusuario.

- Crear un contenedor de imágenes personalizado, donde el usuario pueda subir todas las imágenes que desee de acuerdo al tipo, tamaño o forma. De esta manera, cuando se efectúe el intercambio de mensajes, los mismos se ocultarán dentro de la imagen escogida por este y se estarán intercambiando dos tipos de datos, el mensaje en texto como tal y una foto o ilustración que se desee compartir.

- De acuerdo a las consideraciones o requerimiento del usuario, él mismo podrá personalizar el aplicativo, es decir: configurar el tiempo de bloqueo de la aplicación, escoger el tipo de imagen para ocultar el mensaje, decidir quién puede ver su imagen de perfil y estado, bloquear algún contacto, elegir el tamaño de fuente para la pantalla de chat, entre otros.

- Añadir otros tipos de mensajes a ocultar mediante esteganografía, como videos, archivos de audio, imágenes animadas, documentos de texto, entre otros.

BIBLIOGRAFÍA

AngularJS by Google. HTML enhanced for web apps. (abril 2016). Disponible en: https://angularjs.org/

Aguilar, José. Introducción a SignalR. (20 de marzo de 2012). Disponible en: http://www.variablenotfound.com/2012/03/signalr-iv-hubs.html

Appel, Rachel. Usa SignalR para crear aplicaciones modernas. (octubre de 2012). Disponible en: https://msdn.microsoft.com/es-es/magazine

Basalo, Alberto. ¿Qué es AngularJS? (28 de agosto 2014). Disponible en: http://www.desarrolloweb.com/articulos/que-es-angularjs-descripcion-framework-javascript-conceptos.html

Cordova. Documentation Cordova. (29 de mayo de 2016). Disponible en: https://cordova.apache.org/docs/es/latest/guide/overview/

Cygnet Infotech. Blog Cygnet Infotech. 9 de enero de 2015. Disponible en: http://www.cygnet-infotech.com/blog/phonegap-or-titanium-or-xamarin-which-cross-platform-should-you-choose

INTERNATIONAL STANDARD ISO/IEC 27000. Information technology: Security techniques — Information security management systems. Overview and vocabulary, Third edition, 2014

Mikoluk, Kasia. JQuery vs. JavaScript: ¿Cuál es la diferencia en definitiva? Disponible en: https://blog.udemy.com/jquery-vs-javascript-2-cual-es-la-diferencia-en-definitiva/

MINISTERIO DE LAS TIC. Boletín trimestral del sector TIC - Cifras tercer trimestre de 2016. 03 de enero de 2017. Disponible en: http://colombiatic.mintic.gov.co/602/w3-article-47512.html

MORENO, M. Security ArtWork. 15 de abril de 2010. Disponible en: http://www.securityartwork.es

Moura, William. Esteganografía, el arte de ocultar información. 02 de abril de 2013. Disponible en: http://www.egov.ufsc.br:8080/portal/conteudo/esteganografía-el-arte-de-ocultar-información

NORMA TÉCNICA COLOMBIANA NTC-ISO-IEC 27001. Tecnología de la información. Técnicas de seguridad. Sistemas de Gestión de la seguridad de la Información (SGSI). Requisitos, 2013.

Pérez, Damián. ¿Qué son las bases de datos? (octubre de 2013). Disponible en: http://www.maestrosdelweb.com/que-son-las-bases-de-datos/

WorkLight Webinar Series. Native Web or Hybrid Mobile App Development. Estados Unidos, 2011.